

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra elektroniky

Inovace laboratorních úloh z předmětu MŘS2
Innovation of Laboratory Exercises from the MCS2

Zadání bakalářské práce

Student:

Petr Cyprich

Studijní program:

B0714A060012 Aplikovaná elektronika

Téma:

Inovace laboratorních úloh z předmětu MŘS2
Innovation of Laboratory Exercises from the MCS2

Jazyk vypracování:

čeština

Zásady pro vypracování:

1. Zpracujte zadání a teoretický rozbor jednotlivých laboratorních úloh podle pokynů vedoucího práce.
2. Vypracujte vzorové protokoly k jednotlivým úlohám.
3. Vytvořte podpůrné materiály a návody pro vyučujícího.

Seznam doporučené odborné literatury:

VÁŇA, Vladimír. ARM pro začátečníky. Praha: BEN - technická literatura, 2009. ISBN 978-80-7300-246-6.

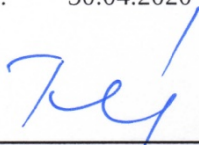
KADLEC, Václav. Učíme se programovat v jazyce C: základ pro programování v C++, C#, Javě, JavaScriptu, PHP a jiných jazycích. Praha: Computer Press, c2002. Programování. ISBN 80-7226-715-9.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Martin Sobek, Ph.D.**

Datum zadání: 01.09.2019

Datum odevzdání: 30.04.2020



doc. Ing. Petr Palacký, Ph.D.
vedoucí katedry

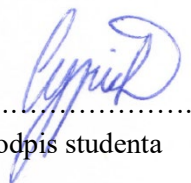


prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: 7. května 2020


.....
podpis studenta

Poděkování

Chtěl bych poděkovat panu Ing. Martinu Sobkovi, Ph.D. za odborné vedení, trpělivost a pomoc při zpracování této práce.

Rád bych také vyjádřil svou vděčnost mé rodině, bez jejíž podpory by tato práce nevznikla.

Abstrakt

Podstatou této bakalářské práce je především obeznámit studenty s náplní předmětu MŘS2, nebrání se však případným potenciálním čtenářům, kterým může rozšířit jejich znalosti v oblasti mikrokontrolérů či poukázat na jejich potenciál a využití v praktických aplikacích. První kapitola této práce slouží k představení aplikačního software od firmy NXP, který studenta při tvorbě těchto úloh bude po celý čas provázet. Následuje podstatná část práce, která je věnovaná důkladnému rozboru jednotlivých laboratorních úloh a jež si klade za cíl seznámit studenta s danou úlohou a klíčovými komponentami, se kterými bude pracovat. Na závěr zde také nalezneme dosažené výsledky, ke kterým se dospělo.

V přílohách se nacházejí jak podrobné manuály, které mají za úkol provést studenta danou úlohou a zjednodušit mu tak práci s konfigurací a nastavením komponent a proměnných v příslušném software, tak také vypracované vzorové protokoly, které pomohou k dosažení správných výsledků.

Klíčová slova

Mikrokontrolér, UART, přerušení, A/D převodník, D/A převodník, PI regulátor, vektorové řízení

Abstract

The essence of this bachelor thesis is primarily to acquaint students with the content of the course MCS2, but does not prevent potential readers, which can extend their knowledge in the field of microcontrollers or point out their potential and use in practical applications. The first chapter of this thesis serves to introduce application software from NXP company, which will accompany the student during the creation of these tasks all the time. The following is a substantial part of the work, which is devoted to a thorough analysis of individual laboratory tasks and which aims to acquaint the student with the task and the key components with which it will work. Finally, we also find the results achieved.

The appendices contain both detailed manuals to guide the student to the task and simplify the work with the configuration and setup of components and variables in the software, as well as elaborated sample protocols that will help to achieve the correct results.

Keywords

Microcontroller, UART, interrupt, A/D converter, D/A converter, PI controller, vector control

Obsah

Seznam použitých symbolů.....	7
Seznam použitých zkratk.....	11
Seznam ilustrací a seznam tabulek.....	13
Úvod	16
1 Aplikační software	17
1.1 Kinetis Design Studio.....	17
1.2 Processor Expert.....	17
1.3 FreeMaster.....	18
2 Rozbor laboratorních úloh.....	20
2.1 Seznámení se s nástrojem Processor Expert – blikání LED pomocí přerušení	20
2.1.1 Popis úlohy	20
2.1.2 Princip generování časových intervalů pomocí komponent PE	20
2.1.3 Dosažené výsledky	21
2.2 Komunikace s uživatelem pomocí nástroje FreeMaster.....	22
2.2.1 Popis úlohy	22
2.2.2 Komunikační rozhraní UART	22
2.2.3 Výpočet parametrů komunikace	24
2.2.4 Výpočet přepony pravoúhlého trojúhelníku.....	24
2.2.5 Dosažené výsledky	25
2.3 Generování harmonických signálů pomocí FreeMasteru.....	26
2.3.1 Popis úlohy	26
2.3.2 D/A převodníky	26
2.3.3 Postup řešení pro vytvoření funkce $\cos(x)$	30
2.3.4 Dosažené výsledky	31
2.4 Měření analogových signálů a jejich zpracování pomocí FreeMasteru	32
2.4.1 Popis úlohy	32
2.4.2 A/D převodníky	32
2.4.3 Výpočet střední a efektivní hodnoty	39
2.4.4 Dosažené výsledky	41
2.5 Návrh a realizace struktury regulace napětí na RC členu pomocí PI regulátoru	43
2.5.1 Popis úlohy	43

2.5.2	PI regulátor	43
2.5.3	Návrh PI regulátoru	45
2.5.4	Realizace PI regulátoru	51
2.5.5	Dosažené výsledky	53
2.6	Programové bloky pro řízení pohonů pomocí FreeMasteru	55
2.6.1	Popis úlohy	55
2.6.2	Vektorové řízení	55
2.6.3	Komplexní souřadnicové systémy	56
2.6.4	Programové bloky	57
2.6.5	Dosažené výsledky	59
3	Vypracované protokoly	61
3.1	Zásady pro vypracování protokolů	61
4	Podpůrné materiály	62
4.1	Funkce jednotlivých komponent vývojového nástroje PE	62
4.2	Dokumentace složitějších funkcí	63
	Závěr	64
	Literatura	65
	Přílohy	67

Seznam použitých symbolů

Symbol	Jednotky	Význam symbolu
$ \mathbf{u}_{\alpha\beta} $	V	Absolutní velikost prostorového vektoru ve SSS
2^n	bit	Počet napěťových hladin
A	-	Konstanta pro normování výpočtu ($A = 1$)
A / U_m	V	Amplituda harmonického signálu
$ah(t)$	-	Akční hodnota (výstup) regulátoru)
BaudRate	baud	Přenosová rychlost
C	F	Kondenzátor (obecně)
d, q	-	Reálná a imaginární složka RSS
Duty	%	Střída signálu
$e(t)$	-	Regulační odchylka (vstup) regulátoru
f	Hz	Frekvence (obecně)
$F_{0(OM)}$	-	Přenos otevřené smyčky dle metody optimálního modulu
$F_0(p)$	-	Přenos otevřené smyčky
$F_I(p)$	-	Přenos integračního členu (zpožďujícího členu 1. řádu)
$F_{R(OM)}$	-	Přenos regulátoru dle metody optimálního modulu
$F_R(p)$	-	Přenos regulátoru
$F_{R(PI)}$	-	Přenos PI regulátoru
$F_S(p)$	-	Přenos soustavy ($F_{S1}(p)$ a $F_{S2}(p)$)
$F_{S1}(p)$	-	Přenos RC článku
$F_{S2}(p)$	-	Přenos A/D a D/A převodníku
K_I	-	Zesílení RC článku
K_I	-	Přepočtená časová konstanta regulátoru

K_2	-	Zesílení obou převodníků
K_2	-	Přepočtená zesilovací konstanta regulátoru
K_R	-	Zesilovací konstanta (proporcionální složky) regulátoru
K_s	-	Zesilovací konstanta setrvačného članku 1. řádu (RC)
K_S	-	Zesilovací konstanta soustavy ($K_S = 1$)
n	bit	Počet bitů
N	-	Bitové slovo
n	-	Počet vzorků harmonického signálu
Offset	V	Stejnoseměrná složka signálu
OSS	-	Orientovaný souřadnicový systém
Phase	°	Fáze signálu
R	Ω	Odpor (obecně)
RSS	-	Rotorový souřadnicový systém
$\sin(x), \cos(x)$	°	Goniometrické funkce, kde x je úhel (argument funkce)
\sin, \cos	°	Velikost sinus a cosinus úhlu ϑ , který prostorový vektor svírá se SSS pro VR2
$\sin\vartheta, \cos\vartheta$	°	Výpočet sinus a cosinus zadaného úhlu pro VR1 ze SSS do RSS
S_{\max}	V	Maximální rozsah převodníku
S_{\min}	V	Minimální rozsah převodníku
SSS	-	Statorový souřadnicový systém
$\text{sum}(k)$	-	Suma pro výpočet integrálu
$\text{sum}(k-1)$	-	Předchozí vypočtená hodnota $\text{sum}(k)$
T_1	s	Časová konstanta RC članku

T_I	s	Integrační časová konstanta
T_R	s	Časová konstanta regulátoru
T_{vz}	s	Vzorkovací perioda
t_{zp}	s	Časové zpoždění
U / V	V	Napětí (obecně)
u_a, u_b, u_c	V	Jednotlivé fáze 3f. signálu
u_d, u_q	V	Reálná a imaginární napěťová složka prostor. vektoru v RSS
U_{om}	V	Vnější omezení signálu
U_p	V	Vstupní požadovaná hodnota
U_s	V	Vstupní skutečná hodnota
U_{vst}	V	Vstupní napětí
U_{vyst}	V	Výstupní napětí
u_x, u_y	V	Reálná a imaginární napěťová složka prostor. vektoru v OSS
u_α, u_β	V	Reálná a imaginární napěťová složka prostor. vektoru ve SSS
V_{1-400}	V	Velikost napětí jednotlivých vzorků
VA	-	Vektorová analýza
V_{ALTH}	V	Alternativní hodnota vysoké úrovně referenčního napětí
V_{ALTHL}	V	Alternativní hodnota nízké úrovně referenčního napětí
V_{AVG}	V	Výsledná střední hodnota napětí harmonického signálu
V_{DDA}	V	Analogové napájení
V_i	V	Vzorek aktuální periody harmonického signálu
V_{i-T}	V	i-tý vzorek předchozí periody harmonického signálu
V_{pp}	V	Mezivrcholová hodnota napětí
$VR1, VR2$	-	Vektorová rotace 1, 2
V_{REF}	V	Referenční napětí (obecně)

Seznam použitých symbolů

V_{REFH}	V	Vysoká úroveň referenčního napětí
V_{REFL}	V	Nízká úroveň referenčního napětí
V_{RMS}	V	Výsledná efektivní hodnota napětí harmonického signálu
V_{SSA}	V	Analogová zem
V_{sum}	V	Součet všech vzorků
$w(t)$	V	Okamžitá hodnota harmonického signálu
x, y	-	Reálná a imaginární složka OSS
α, β	-	Reálná a imaginární složka SSS
σ	%	Velikost překmitu
τ_1	s	Největší časová konstanta v obvodu, kterou budeme kompenzovat (RC článku)
τ_m	s	Rychlost regulace
τ_r	s	Doba regulace
τ_σ	s	Součet malých časových konstant soustavy (A/D a D/A převodníků)
ω_m	rad/s	Úhlová rychlost rotoru

Seznam použitých zkratk

Zkratka	Význam
μP	Mikropočítač, mikrokontrolér
ADC	Analog to Digital Converter (A/D převodník)
BaudRate	Bits of Actual Usable Data Rate (přenosová rychlost)
BDM	Background debug mode (režim ladění na pozadí)
BRD	Baud Rate Divisor (registr děličky přenosové rychlosti)
BRFA	Baud Rate Fine Adjust (registr pro jemné nastavení přenosové rychlosti)
CAN	Controller Area Network (datová sběrnice místní sítě, protokol multiplexní sériové komunikace)
COM	Communication (komunikační port)
CPU	Central Processing Unit (centrální procesorová jednotka)
ČMT2	Číslicová a mikroprocesorová technika 2
DAC	Digital to Analog Converter (D/A převodník)
DMA	Direct Memory Acces (přímý přístup do paměti)
FET	Field Effect Transistor (tranzistor řízený elektrickým polem)
FLASH	Flash memory (RAM) (energeticky nezávislá polovodičová paměť)
FM	FreeMaster
IDE	Integrated Development Environment (integrované vývojové prostředí)
JTAF	Join Test Action Group (skupina pro označení standardu pro výrobu integrovaných obvodů (IC))
KDS	Kinetis Design Studio
LED	Light Emitting Diode (světlo vyzařující dioda)
MŘS2	Mikroprocesorové řídicí systémy II
NXP	Next Experience (dříve Philips Semiconductors)
OM	Optimum Modulus (metoda optimálního modulu)
OpenSDA	Open-standard Serial and Debug Adapter (sériový a ladící adaptér)

PE	Processor Expert
PI	Proportional Integral controller (proporcionálně integrační regulátor)
PIT	Periodic Interrupt Timer (časovač periodického přerušení)
PWM	Pulse Width Modulation (pulzní šířková modulace)
RC	Článek tvořený R a C prvky
RxD	Receive Data pin (pin pro příjem dat)
SAR	Successive Approximation Register (registr postupných aproximací)
SO	Symmetrical Optimum (metoda symetrického optima)
SRAM	Static Random Access Memory (statická paměť)
TSV	Time Start Value (registr počáteční hodnoty časovače)
TxD	Transmitter Data pin (pin pro vysílání dat)
UART	Universal Asynchronous Receiver Transmitter (univerzální asynchronní přijímač a vysílač)

Seznam ilustrací a seznam tabulek

Číslo ilustrace	Název ilustrace	Číslo stránky
1	Vývojová deska s Arm Cortex M4 (MK64FN1M0VLQ12)	16
2	Zobrazení vývojového prostředí KDS	17
3	Zobrazení hlavních položek vývojového nástroje PE v KDS	18
4	Zobrazení vývojového nástroje FM	19
5	Umístění LED (PTB2 – PTB7) na vývojové desce	20
6	Zobrazení průběhu výstupního napětí přiváděného na vstup LED	21
7	Komunikace MCU s uživatelem pomocí UART přes OpenSDA rozhraní	22
8	Zobrazení 1 paketu přenášených dat pomocí komunikačního protokolu UART (podle [1])	23
9	Snímek okna sledování proměnných pořízených z FreeMasteru pro výpočet přepony pravoúhlého trojúhelníku pomocí odhadů s hodnotami odvěsen $A = 31$, $B = 18$, výsledek $C = 35$	25
10	Umístění D/A převodníků (DAC0, DAC1) s vyznačeným duálním operačním zesilovačem AD823A pro zvětšení rozsahu na vývojové desce	26
11	Blokové schéma D/A převodníku (podle [2])	26
12	Blokové schéma DAC modulu (podle [2])	27
13	Obvodové zapojení pro získání stabilizovaného napájecího napětí 3,3 V [3]	28
14	Obvod v zapojení s duálním operačním zesilovačem pro zvětšení rozsahu obou D/A převodníků [3]	29
15	Aproximace funkce $\cos(x)$ v bodě $x_0 = 0$ pro $n = 6$ [4]	30
16	Zobrazení generovaných funkcí sinus a cosinus na osciloskopu pro $f \approx 50$ Hz	31
17	Umístění dvojice A/D převodníků (A1-A2, A3-A4) s vyznačeným duálním operačním zesilovačem AD823A pro zvětšení rozsahu na vývojové desce	32

18	Blokové schéma A/D převodníku (podle [2])	32
19	Převod analogového signálu na digitální (diskrétní) (podle [5])	33
20	Blokové schéma ADC modulu (podle [2])	34
21	Blokový diagram obvodu Sample and hold [6]	35
22	Převod pomocí metody postupné aproximace (podle [2])	36
23	Obvodové zapojení pro získání stabilizovaného referenčního napětí 1,25 V pro rozdílový zesilovač [3]	37
24	Obvodové zapojení pro získání stabilizovaného referenčního napětí úrovně „H“ pro A/D převodníky [3]	37
25	Obvodové zapojení s operačním zesilovačem pro zvětšení rozsahu A/D převodníku	38
26	Rozdělení periody průběhu na stejný počet částí	39
27	Snímek generovaných obdélníkových průběhů ve FreeMasteru s parametry: adc_napeti_ch0: f = 50 Hz, Amp = 4 Vpp, Offset = 2 V, Phase = 0°, Duty = 70 % adc_napeti_ch1: f = 50 Hz, Amp = 2 Vpp, Offset = 1 V, Phase = 0°, Duty = 20 %	41
28	Snímky generovaných funkcí z osciloskopu (vlevo) a ve FreeMasteru (vpravo)	42
29	Umístění A/D (A1-A2, A3-A4) a D/A (DAC0, DAC1) převodníků na vývojové desce	43
30	Blokové zapojení složek PI regulátoru [7]	43
31	Zobrazení jednotkového skoku na vstupu PI regulátoru pro zjištění přechodové charakteristiky (podle [7])	44
32	Regulátor bez vnitřního omezovače s vyznačením zpoždění reakce t_{zp} (podle [7])	44
33	Regulátor s vnitřním omezovačem (bez zpoždění, $t_{zp} = 0$) (podle [7])	44
34	Přechodová charakteristika PI regulátoru s odezvou na skokovou hodnotu napětí [7]	45
35	Blokové zapojení PI regulátoru k RC filtru [8]	45
36	Blokové zapojení regulačního obvodu	46
37	Přechodové charakteristiky dle metody SO bez a s vlivem filtru s vyznačením charakteristických	50

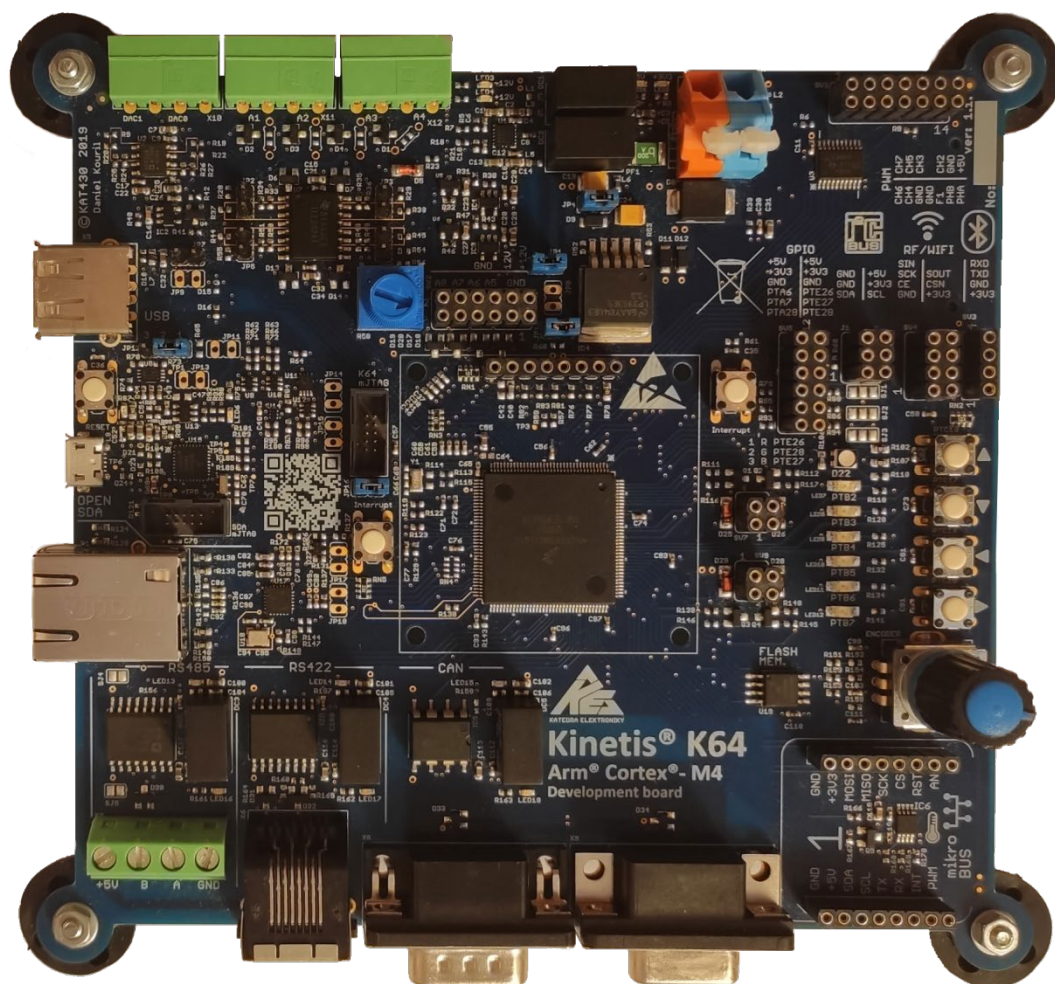
	parametrů [9]	
38	Vývojový diagram PI regulátoru realizovaný pomocí funkcí v číslicové podobě (podle [10])	52
39	Snímek odezvy navrženého PI regulátoru na vstupní pravoúhlý signál ve FreeMasteru s vyznačením parametrů a okamžité reakce/změny, $K_R = 13,15$, $T_R = 2,63$ ms	53
40	Snímek regulace RC članku připojeného na výstup PI regulátoru ve FreeMasteru se změnou hodnoty konstanty zesílení $K_R = 10,4$, $T_R = 2,63$ ms, $V_{pp} = 1$ V, $f = 20$ Hz	54
41	Snímky průběhu akčního zásahu (vlevo) a požadované a skutečné hodnoty napětí (vpravo) při regulaci RC članku připojeného na výstup PI regulátoru zaznamenaného z osciloskopu, $K_R = 13,15$, $T_R = 2,63$ ms, $V_{pp} = 1$ V, $f = 20$ Hz	54
42	Zobrazení základních komplexních souřadnicových systémů s transformací napětí (podle [11])	56
43	Blokové zapojení jednotlivých funkčních bloků (podle [12])	57
44	Snímek vektorové rotace 2fázového signálu ze SSS (u_α , u_β) do RSS (d, q) dle úhlu $\varphi = 3,14$ rad ve FreeMasteru	59
45	Zobrazení vektorové rotace z RSS (d, q) do OSS (x, y) dle úhlu získaného pomocí VA ve FreeMasteru, $\varphi = \pi$ rad	60
46	Zobrazení používaných přístrojů k měření a testování aplikací	64

Číslo tabulky	Název tabulky	Číslo stránky
1	Výpočet parametrů rozhraní UART pro 5 přenosových rychlostí	25
2	Srovnání parametrů přechodových charakteristik dle použitých optimalizačních metod	49

Úvod

Cílem této bakalářské práce bylo navrhnout a zrealizovat 6 laboratorních úloh zabývajících se využitím mikropočítače řady K64 od společnosti NXP v praktických aplikacích. Jedná se konkrétně o mikropočítač s označením MK64FN1M0VLQ12, který disponuje výkonným jádrem Cortex M4 s harvardskou architekturou, u které je fyzicky oddělená paměť dat a programu. Mezi hlavní přednosti jádra M4 patří dosažení provozní frekvence až 120 MHz a statická paměť SRAM o velikosti 256 kB, 16kanálový řadič DMA pro přímý přístup do paměti a nízká spotřeba energie potřebná a užitečná v nízkoeenergetických aplikacích.

Mikropočítač, který budeme v našich aplikacích využívat, je vlastně jádro Cortex M4 obohacené o periferní obvody, které poskytují daleko širší využití a nacházejí uplatnění v oborech zabývajících se řízením a regulací systémů a zařízení. V následujících laboratorních úlohách se budeme věnovat pouze hrstce periférií, které tento mikropočítač nabízí. Jedná se především o 16bit. časovače, 16bit. A/D a 12bit. D/A převodníky.

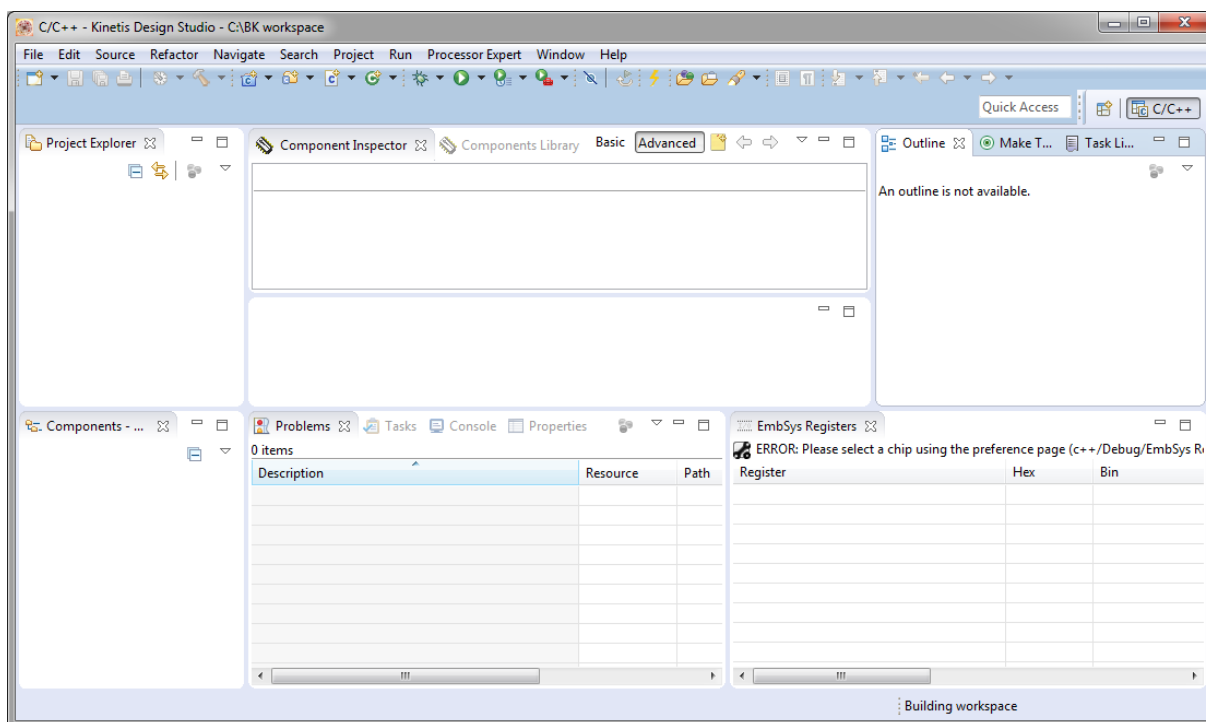


Obr. 1: Vývojová deska s Arm Cortex M4 (MK64FN1M0VLQ12)

1 Aplikační software

1.1 Kinetis Design Studio

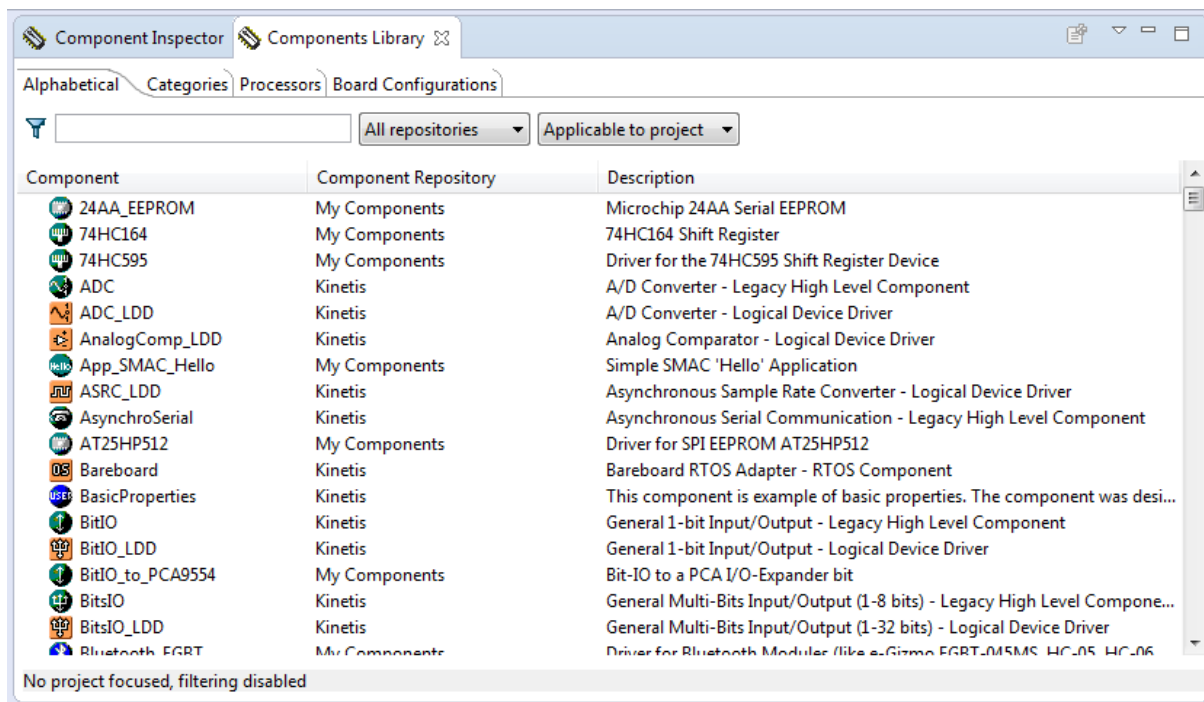
Kinetis Design Studio, dále již KDS, je integrované vývojové prostředí pro mikropočítače od společnosti NXP, které nabízí velké množství uživatelských funkcí. Mezi jeho hlavní přednosti patří jednoduchost a spolehlivost bez omezení velikosti psaného kódu. Toto prostředí nám poskytuje všechny potřebné nástroje pro naše aplikace. Jedná se o počítačový software od stejnojmenné společnosti vyrábějící námi používaný mikropočítač řady K64, který však již přišel o podporu a byl plně nahrazen IDE prostředím MCUXpresso.



Obr. 2: Zobrazení vývojového prostředí KDS

1.2 Processor Expert

Nedílnou součástí tohoto prostředí, bez kterého by náš vývoj aplikací byl bez předešlých znalostí mnohem obtížnější a v mnoha ohledech výrazně komplikovanější, je vývojový nástroj *Processor Expert*, dále jen PE, poskytující nám nespočet předpřipravených knihoven s funkcemi příslušející jednotlivým periferním obvodům, nazývaných jako komponenty. PE slouží jako doplněk prostředí KDS a je možné ho již integrovaného v KDS stáhnout z oficiálních stránek společnosti NXP.



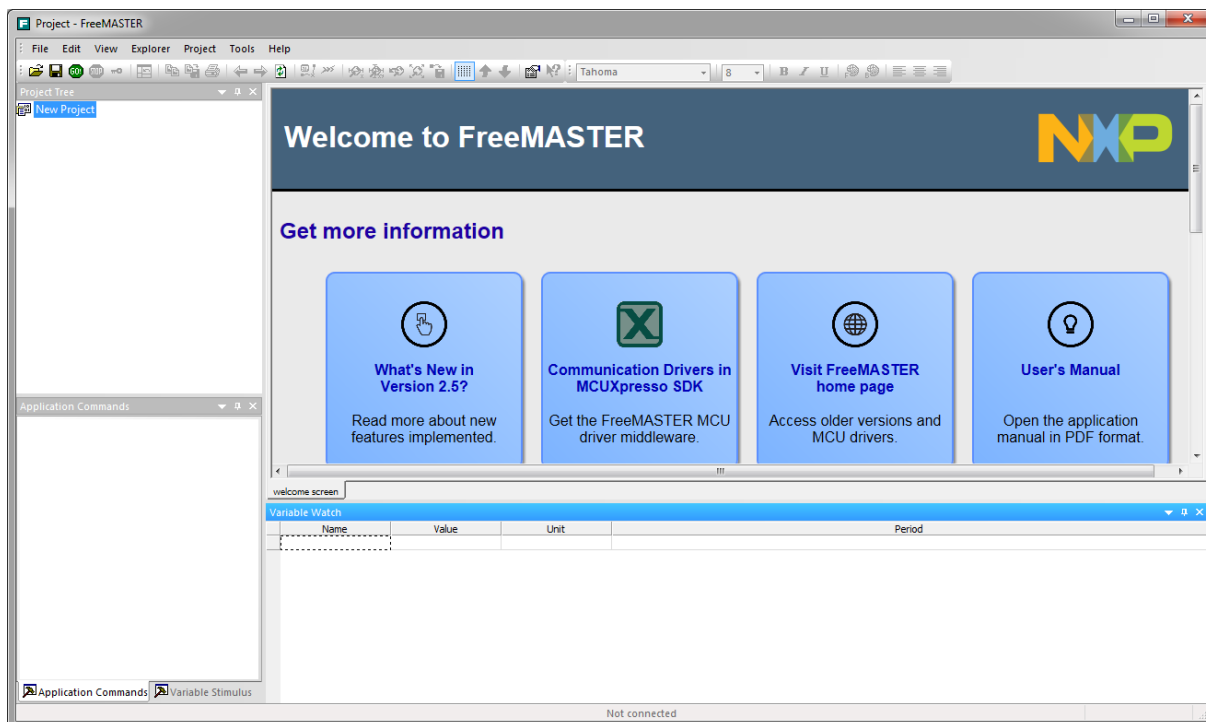
Obr. 3: Zobrazení hlavních položek vývojového nástroje PE v KDS

PE se skládá ze tří hlavních částí:

- *Components Library*, pomocí níž lze vybírat a přidávat komponenty do našeho adresáře komponent, který je také jeho součástí.
- *Component Inspector*, ve které je možné konfigurovat jednotlivé komponenty naimportované v adresáři komponent (*Components*).
- *Components*, ve kterém jsou naimportované zvolené komponenty z *Component Library*.

1.3 FreeMaster

Řešení laboratorních úloh se zakládá na práci s *FreeMasterem*, dále již FM, pomocí něhož má uživatel možnost analyzovat programový kód a reagovat tak na vzniklé problémy. Jedná se o grafický nástroj, který slouží ke čtení nebo zápisu do paměti pomocí sériového portu (COM) nebo dle vyhrazeného komunikačního protokolu (BDM, JTAG či CAN). V našich úlohách budeme využívat komunikační rozhraní UART, pomocí kterého bude FM komunikovat s mikropočítačem vykonávající zvolený program. Tento nástroj využívá aplikační binární soubor s koncovkou *.elf*, který se vytváří při sestavování programu v KDS, pro zjištění informací o umístění globálních proměnných v cílové paměti. Tyto globální proměnné je *FreeMaster* pomocí svého prostředí schopen zobrazovat v reálném čase. [13]



Obr. 4: Zobrazení vývojového nástroje FM

FM se skládá ze čtyř hlavních částí:

- *Project Tree* (strom projektu), který obsahuje vytvořený projekt se všemi jeho dílčími částmi.
- *Application Command* (aplikační příkazy), ve kterém můžeme vytvářet příkazy, které bude FM vykonávat.
- *Detail View area* (okno podrobného zobrazení, v místě uvítací obrazovky Welcome screen) zobrazuje dynamicky se měnící hodnoty proměnných dílčích bloků FM přidáných do projektu, např. *Recorder*.
- *Variable Watch* (sledování proměnných) nám poskytuje skutečné okamžité hodnoty proměnných a umožňuje nám měnit jejich velikost, pokud je to v nastavení daných proměnných povoleno.

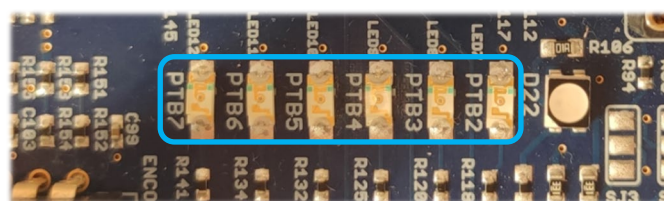
V těchto laboratorních úlohách budeme do projektu přidávat blok nazývaný *Recorder* (záznamník). *Recorder* periodicky čte hodnoty proměnných a vykresluje je v reálném čase. Jeho princip je založen na vzorkování proměnných a ukládání těchto navzorkovaných hodnot do vyrovnávací paměti mikropočítače, ze které jsou po dosažení nastavené max. hodnoty počtu vzorků takto navzorkovaná data odeslána do programu *FreeMaster*, v jehož okně podrobného zobrazení se graficky zobrazují. Jednotlivé parametry *FreeMasteru* se nastavují jak v samotném programu, tak také v komponentě PE v prostředí KDS. [13]

2 Rozbor laboratorních úloh

2.1 Seznámení se s nástrojem Processor Expert – blikání LED pomocí přerušení

2.1.1 Popis úlohy

Tato počáteční úloha, nazvaná také jako „nultá“ úloha, si klade za cíl seznámit uživatele s vývojovým nástrojem *Processor Expert*, jeho možnostmi nastavení a využití ke zjednodušení tvorby programu. Příklad využití tohoto nástroje bude demonstrován na blikání LED. Časový interval blikání je zde vytvořen komponentou *Wait* a následně také za pomoci přerušení časovače komponentou *TimerInt*. Úloha nás provede postupem založení nového projektu s výběrem nástroje PE v KDS, vytvořením nových zdrojových a hlavičkových souborů v adresáři *Sources*, volbou a nastavením příslušných komponent v nástroji PE, včetně podrobné konfigurace samotného mikroprocesoru, dále již CPU. Závěr se zabývá nejen sestavením, ale také laděním (debuggovaním) vytvořeného programu, které je spojeno s jeho zavedením do FLASH paměti mikropočítače.



Obr. 5: Umístění LED (PTB2 – PTB7) na vývojové desce

2.1.2 Princip generování časových intervalů pomocí komponent PE

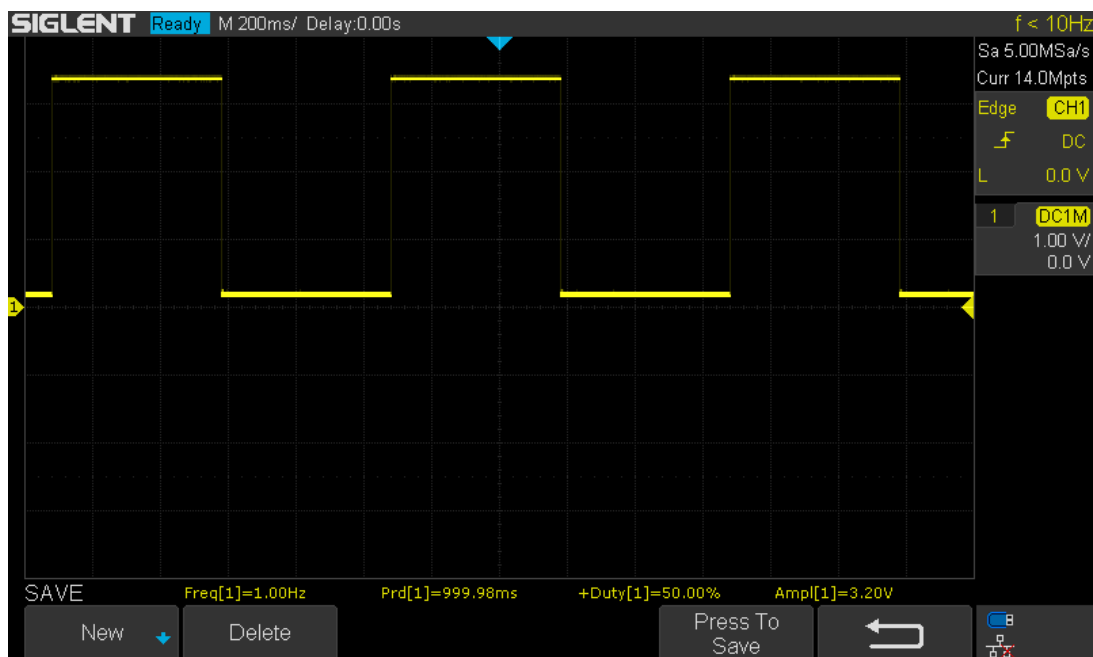
Časové intervaly můžeme generovat dvěma odlišnými způsoby. Jedním z nich je zavedení čekající rutiny, kterou zaměstnáme CPU na požadovanou dobu, po jejíž uplynutí dojde k rozsvícení/zhasnutí připojené LED. Pro tento účel využijeme komponentu *Wait*, která nám mimo jiné nabízí funkce s již předdefinovanými cykly čekání.

Následuje generování přerušení pomocí časovače, které nastává periodicky dle nastavené periody přerušení. Pomocí jeho vlastní komponenty *TimerInt* v *Processor Expert* si můžeme zvolit libovolný zdroj přerušení časovače, který nabízí zvolený mikropočítač. Každá komponenta vygenerovaná pomocí PE pracující na principu žádosti přerušení obsahuje ve zdrojovém souboru s názvem *Events.c* vlastní podprogram pro obsluhu přerušení. U časovače je tento podprogram nazvaný jako *Tlx_OnInterrupt*. V rámci vykonání tohoto podprogramu se provede změna stavu log. úrovně na pinu, ke kterému je připojena LED.

Mikropočítač obsahuje různé druhy časovačů periodického přerušení, které mj. slouží pro spouštění A/D převodů. Ve své práci využívám časovače modulu PIT (*Periodic Interrupt Timer*). Tyto moduly časovače disponují až 32bitovým rozlišením. Hlavní registr PIT_LDVALn (*Timer Load Value Register*) časovače PIT slouží k nastavení počáteční hodnoty časování. Od této hodnoty se poté časovač dekrementuje. Jakmile dosáhne nuly, vygeneruje přerušení. Po vykonání obsluhy přerušení se opět začne dekrementovat od počáteční přednastavené hodnoty. Nástroj PE nám jednotlivá nastavení registrů převádí do grafické podoby, která nám práci s nimi výrazně usnadňuje. [14]

2.1.3 Dosažené výsledky

Následující snímek zobrazuje obdélníkový signál přiváděný na připojenou LED se střídou signálu 50%. Tento signál s periodou 1 s je generován s pomocí přerušení časovače, jehož přerušení je vyvoláváno každých 500 ms. Za tento čas se voláním funkce pro negaci stavu *LED1_Neg*, která je nabízená komponentou *LED*, LED periodicky rozsvítí nebo zhasne.



Obr. 6: Zobrazení průběhu výstupního napětí přiváděného na vstup LED

Příklady použití funkcí

Generování časových intervalů pomocí komponenty *Wait*:

```
void loop(void) {           /* blikani LED1 pomoci casoveho zpozdeni */
    LED1_Neg();              /* negace LED1 */
    WAIT1_Waitms(200);       /* cekej 200 ms */
}
```

Generování přerušení pomocí časovače:

```
void isr_pit0(void) {        /* blikani LED2 pomoci preruseni casovace */
    LED2_Neg();              /* negace LED2 */
}
```

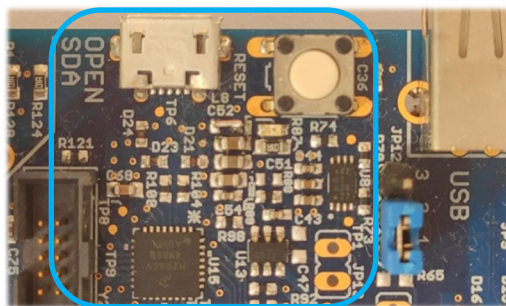
Funkce nacházející se v událostech přerušení časovače (*Events.c*):

```
void TI1_OnInterrupt(void)
{
    isr_pit0();              /* blikani LED2 pomoci preruseni casovace */
}
```

2.2 Komunikace s uživatelem pomocí nástroje FreeMaster

2.2.1 Popis úlohy

Cílem laboratorní úlohy je představit uživateli grafický nástroj *FreeMaster*, jeho možnosti a využití při vývoji programu. Tento nástroj si právem zasloužil pozornost svou jednoduchostí a přístupem k uživateli. Díky jeho možnostem, které nabízí, je vývoj aplikací mnohem jednodušší, jak uvidíme dále. Nabízí nespočet možností, kterým se budeme věnovat v následujících úlohách. Nezůstaneme však pouze v teoretické rovině, protože postup popsáný v tomto projektu nám s pomocí tohoto nástroje pomůže s praktickým řešením aplikace pro výpočet přepony pravoúhlého trojúhelníku s použitím odhadů. Výsledky jednotlivých výpočtů budeme schopni pozorovat v okně *FreeMasteru* pro sledování stavu proměnných, jehož název nese označení *Variable Watch*. Tento nástroj bude s našim mikropočítačem komunikovat přes komunikační rozhraní UART, u kterého si vyzkoušíme vypočítat jednotlivé parametry přenosových rychlostí, mezi nichž se řadí registry BRD a BRFA. Registr BRD slouží jako dělička přenosové rychlosti, zatímco BRFA se používá pro jemné doladění k dosažení požadované rychlosti. Pro komunikaci mikropočítače s uživatelem pomocí UART se bude využívat rozhraní OpenSDA, které se nachází na vývojové desce. Podrobnou konfiguraci komponenty nástroje FM spolu s nastavením jeho komunikačního rozhraní v *Processor Expert* si také podrobně ukážeme. Jak se s tímto nástrojem pracuje a jaký je správný postup pro přidávání proměnných do *Variable Watch* tak, aby nám plnil úlohu, kterou po něm požadujeme, nám objasní samostatná kapitola rovněž obsažená v postupu pro tvorbu tohoto projektu.



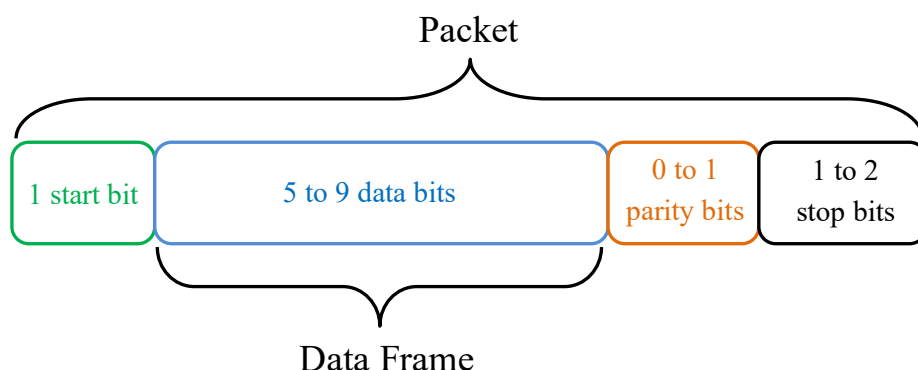
Obr. 7: Komunikace MCU s uživatelem pomocí UART přes OpenSDA rozhraní

2.2.2 Komunikační rozhraní UART [1, 15]

U mikropočítačů se můžeme setkat s dvěma druhy komunikace: sériová a paralelní. V případě sériového přenosu se data mezi dvěma zařízeními účastníci se komunikace přenáší bit po bitu. Paralelní přenos dat dosahuje daleko vyšších rychlostí, protože se všechny bity tvořící data přenášejí současně. Avšak požadavky na vyšší počet vodičů a přídavných obvodů tento způsob přenosu činí velmi nákladným. Z těchto důvodů se hojně využívá sériového přenosu vyžadující pouze dva vodiče pro příjem a přenos dat. Tento přenos se dále dělí na synchronní, který využívá společný hodinový signál pro synchronizaci přenosu mezi vysílacím a přijímacím zařízením, a asynchronní, u kterého je tento hodinový signál nahrazen speciálními bity indikující parametry probíhajícího přenosu.

UART (*Universal Asynchronous Receiver and Transmitter*), jak již z názvu vyplývá, je zařízení sloužící pro asynchronní přenos dat. Mikropočítač, s kterým pracujeme, obsahuje 6 UART modulů, z nichž pět je vyvedených na vývojové desce.

Zařízení, která se účastní této komunikace, obsahují dva piny nazývající se *TxD* a *RxD*, pomocí nichž se požadovaná data přenáší. Následující *obr. 8* popisuje princip přenosu dat s pomocí tohoto rozhraní.



Obr. 8: Zobrazení 1 paketu přenášených dat pomocí komunikačního protokolu UART (podle [1])

Data, která jsou určená k přenosu, přicházejí na datovou sběrnici z centrální procesorové jednotky, tedy z CPU. Zařízení určené pro vysílání tato data na svých vstupech přijímá v paralelní podobě. Přijatá data se následně převádějí na sériovou formu, ke které se během tohoto převodu přiřazují speciální bity, jenž vynahrazují chybějící synchronizační signál. Prostřednictvím těchto speciálních bitů je zařízení schopné rozpoznat začátek a konec uskutečňujícího se přenosu. Takto modifikovaná data tvoří paket, který se přenáší z *TxD* pinu vysílacího zařízení na pin *RxD* zařízení přijímacího.

Datový paket je vybaven *start bitem*, který při přenosu převádí stav datové přenosové linky z log. 1 na úroveň log. 0 po dobu jednoho hodinového cyklu. Přijímací zařízení detekuje změnu stavu linky a začne číst skutečná data v rozsahu 5-9 bitů. Rychlost čtení dat závisí na předem nastavené přenosové rychlosti.

Paritní bit slouží k ověření správnosti přenosu. Přijímací zařízení po skončení čtení dat z datového rámce ověří sudost či lichost přenesených dat. Jestliže tedy vysílací zařízení nastavilo parity bit na log. 1 (lichá parita), musí součet bitů v datovém rámci s hodnotou 1 odpovídat lichému počtu. V případě sudé parity (log. 0) musí být výsledná hodnota sudá. Pokud v obou případech panuje shoda, přijímací zařízení vyhodnotí převod jako úspěšný. Bude-li se hodnota oproti parity bitu lišit, rozpozná, že došlo ke změně bitů v datovém rámci a tak k chybě přenosu. Nastavení parity bitu není při přenosu dat podmínkou. Tento paket je ukončen bitem nazývajícím se *stop bit*, který signalizuje konec probíhajícího přenosu dat převedením stavu přenosového vedení z log. 0 na log. 1 po dobu 1-2 bitů.

Přijímací zařízení čte tento přenášený paket na svém pinu *RxD* bit po bitu a převádí čtená data zpět na paralelní formu, ze které odstraňuje speciální bity. Paralelní data jsou následně přivedena na datovou sběrnici přijímacího zařízení, ze které jsou k dispozici pro další využití.

2.2.3 Výpočet parametrů komunikace [16]

Následující parametry rozhraní UART se budou počítat pro 5 odlišných přenosových rychlostí: 9600, 19200, 38400, 57600, 115200 [baud] dle následujících vztahů.

Pro výpočet přenosové rychlosti uplatníme vzorec:

$$BaudRate = \frac{UART\ module\ clock}{16 \cdot \left(BRD + \left(\frac{BRFA}{32} \right) \right)} \quad 2.2.1$$

UART module clock ... UART využívá frekvenci z *Bus clock*, tedy 60 MHz.

Z výše uvedeného vztahu vyplývá, že pro nastavení rychlosti komunikace je nezbytné znát parametry *BRD* a *BRFA* děličky přenosové rychlosti.

BRD (*Baud rate divisor*) vypočítáme pomocí vztahu:

$$BRD = \frac{UART\ module\ clock}{16 \cdot BaudRate} \quad 2.2.2$$

Stejný postup aplikujeme na výpočet *BRFA* (*Baud rate fine adjust*) dle vztahu:

$$BRFA = \left(\frac{UART\ module\ clock}{16 \cdot BaudRate} - BRD \right) \cdot 32 \quad 2.2.3$$

2.2.4 Výpočet přepony pravoúhlého trojúhelníku [16]

Pro výpočet přepony použijeme známý vztah odvozený z Pythagorovy věty:

$$c = \sqrt{a^2 + b^2} \quad 2.2.4$$

K výpočtu odmocniny v našem programu budeme využívat následujících vztahů pro odhad:

$$odhad_{k-1} = \frac{|a| + |b|}{2} \quad 2.2.5$$

$$odhad_k = \frac{\frac{a^2 + b^2}{odhad_{k-1}} + odhad_{k-1}}{2} \quad 2.2.6$$

Pro dosažení správné hodnoty provedeme výpočet odhadu dle výše uvedených vzorců vícekrát za sebou.

2.2.5 Dosažené výsledky

Tab. 1 nám ukazuje vypočtené hodnoty registrů rozhraní UART pro různé přenosové rychlosti. Můžeme vidět, že zatímco u prvních 3 přenosových rychlostí nám díky přesně vypočteným hodnotám registrů BRD a BRFA vychází nulová odchylka, u zbylých rychlostí se již při zadávání těchto vypočtených parametrů do nastavení subkomponenty *UART* vývojového nástroje *Processor Expert* dopouštíme odchylek přesahující 0,01% požadované rychlosti, které vznikají vlivem zaokrouhlování hodnot registru BRFA na celá čísla, kdy nejvíce se lišíme u nejvyšší přenosové rychlosti 115200 baud, přičemž dosahujeme nepřesnosti 0,03199%.

Tab. 1: Výpočet parametrů rozhraní UART pro 5 přenosových rychlostí

BaudRate (baud)	BRD	BRFA	BaudRate PE (baud)	Odchylka (%)
9600	390	20	9600	0
19200	195	10	19200	0
38400	97	21	38400	0
57600	65	3	57609,217	0,016
115200	32	18	115163,148	0,03199

Následující snímek na obr. 9 nám zobrazuje okno grafického nástroje *FreeMaster*, ve kterém je dle zadání velikostí odvěsen A a B vypočtena přepona pomocí jednotlivých odhadů. Zjistil jsem, že k vypočtení přesné hodnoty odvěsny nám postačují pouze první 3 odhady. Výpočet těchto odhadů byl prováděn za pomoci bitového posunu, který nám nahrazoval dělitele dvou (vztah 2.2.6).

Name	Value	Unit	Period
A	31	DEC	100
B	18	DEC	100
C	35	DEC	100
odhad0	24	DEC	100
odhad1	38	DEC	100
odhad2	35	DEC	100
odhad3	35	DEC	100

Obr. 9: Snímek okna sledování proměnných pořízených z *FreeMasteru* pro výpočet přepony pravoúhlého trojúhelníku pomocí odhadů s hodnotami odvěsen $A = 31$, $B = 18$, výsledek $C = 35$

Příklad výpočtu přepony pravoúhlého trojúhelníku

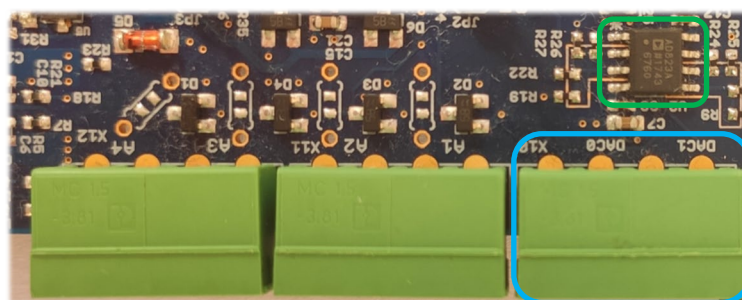
```
void loop(void){ /* vypocet prepony pravouhleho trojuhelniku */
suma_A2B2 = (A*A)+(B*B); /* nasobeni 16bit. promennych = 32bit vysledek */
odhad0 = (abs(A)+abs(B)>>1); /* 1. odhad => (|a|+|b|)/2 */
odhad1 = ((suma_A2B2/odhad0)+odhad0)>>1; /* 2. odhad */
odhad2 = ((suma_A2B2/odhad1)+odhad1)>>1; /* 3. odhad */
odhad3 = ((suma_A2B2/odhad2)+odhad2)>>1; /* 4. odhad */
C=odhad3; /* vysledny odhad */
FMSTR1_Poll(); /* funkce pro udrzeni komunikace s FreeMasterem */
}
```

Velmi důležité je v tomto případě neopomenout funkci *Poll* komponenty *FreeMaster*, bez které by nám tento nástroj nepracoval správně.

2.3 Generování harmonických signálů pomocí FreeMasteru

2.3.1 Popis úlohy

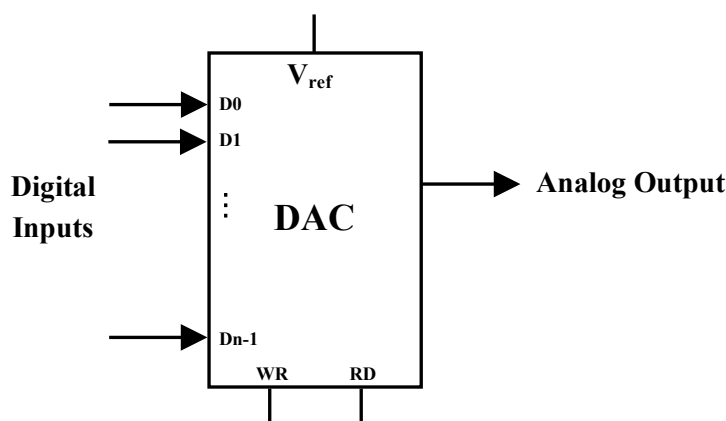
Tato úloha je věnována D/A převodníkům, jejich principu a využití při tvorbě aplikací. Vyzkoušíme si generování harmonických funkcí sinus a cosinus, které budeme realizovat dvěma odlišnými způsoby, tedy pomocí tabulky a Taylorova polynomu 6. řádu. Ukážeme si, jak takto generované funkce můžeme zobrazit ve *FreeMasteru* a s pomocí D/A převodníků, jejichž nastavení a podrobnému rozboru se bude zabývat samostatná kapitola, také na osciloskopu. V této úloze se rovněž seznámíme s další funkcí nabízenou grafickým nástrojem, kterou bude záznamník (*Recorder*), jenž je zcela nezbytný pro zobrazování jakýchkoli průběhů pomocí *FreeMasteru*.



Obr. 10: Umístění D/A převodníků (DAC0, DAC1) s vyznačeným duálním operačním zesilovačem AD823A pro zvětšení rozsahu na vývojové desce

2.3.2 D/A převodníky

D/A převodníky jsou zařízení, která převádějí vstupní datové slovo na spojitý analogový signál definovaný v každém časovém okamžiku.



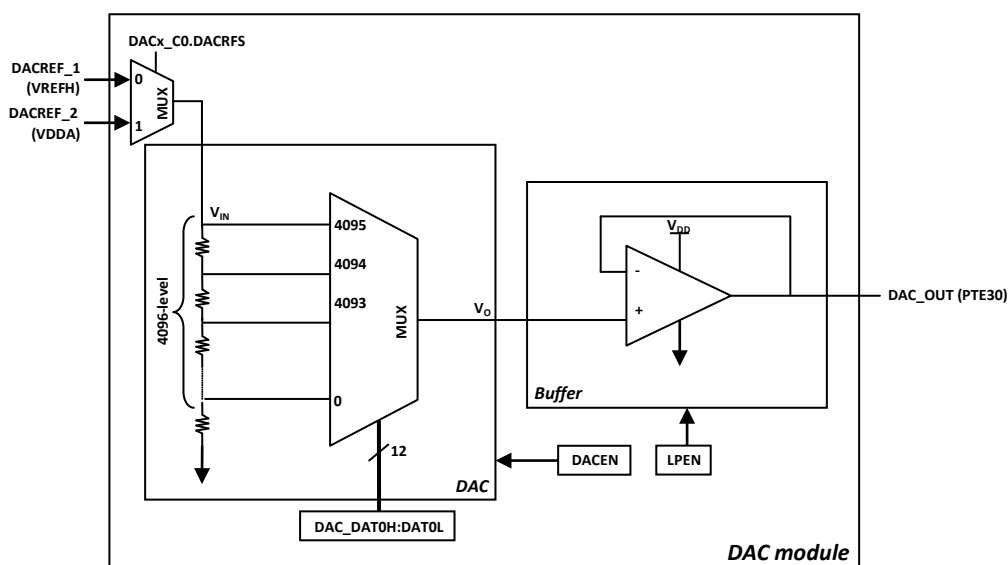
Obr. 11: Blokové schéma D/A převodníku (podle [2])

Na vstup převodníku se přivádějí digitální data ve formě jednotlivých bitů, jejichž počet závisí na rozlišení převodníku. V případě, že tedy disponujeme 12bit. převodníkem, na vstup můžeme přivést až 12 bitů. Zvolené rozlišení převodníku nám také stanovuje formát analogového výstupu, jinými slovy, počet napěťových hladin určené vztahem 2^n , kde n je počet bitů přiváděných na vstup.

12bit. D/A převodník vytváří $2^{12} = 4096$ diskrétních napěťových hladin. V praxi se vyskytují převodníky s rozlišením 8, 10, 12 a 16 bitů, z nichž poslední skupina patří mezi ty nejdražší. Hlavní využití nacházejí při reprodukci digitálního zvuku.

Popis modulu DAC

Mikrokontrolér obsahuje dvojici D/A převodníků s rozlišením 12 bitů, které jsou vyvedeny na vývojové desce. Blokové schéma modulu DAC (*Digital Analog-converter*) na obr. 12 přibližuje princip činnosti tohoto převodu.



Obr. 12: Blokové schéma DAC modulu (podle [2])

V praxi existují dva druhy převodníků sloužící pro převod vstupního číslicového signálu ve formě datového slova na analogový signál spojitý v čase. Mezi nejběžnější se řadí D/A převodník s žebříčkovou sítí rezistorů R-2R. Používá se z důvodu dosažení mnohem vyššího stupně přesnosti oproti následujícímu druhu, tedy převodníku s váhovými rezistory. Modul převodníku zobrazený na obr. 12 se skládá z žebříčkové sítě tvořené rezistory a analogového multiplexeru. Pro činnost převodníku je nezbytné přivedení hodinového signálu, který je pro převodník DAC0 zajištěn pomocí kontrolního registru *SIM_SCGC6* (*System Clock Gating Control Register 6*) přivedením log. 1 na 31. pin. Hodinový signál pro tento převodník může být také přiveden pomocí registru *SIM_SCGC2*, který jako jediný tento signál přivádí i pro druhý převodník DAC1. [14]

Každý z převodníků má vlastní kontrolní registr s názvem *DACx_C0*, pomocí kterého se chod těchto převodníků musí povolit přivedením log. 1 na příslušný bit v bitovém poli. Jak můžeme vidět výše v blokovém schématu, pomocí tohoto registru, přesněji pomocí bitového pole *DACRFS* tohoto registru, se také vybírá 1 z referenčních napětí. K dispozici je *DACREF_1* jako V_{REFH} (1,13 V – V_{DDA}) a *DACREF_2* rovnající se hodnotě V_{DDA} (1,71 – 3,6 V). Datové registry *DACx_DATnH* a *DACx_DATnL* slouží pro uchování 12bitové digitální hodnoty určené pro převod na analogový signál. Zatímco u *DATnL* se používají všechny bity bitového pole pro binární data, u *DATnH* jsou pro data určeny pouze 4 nejnížší bity. [14]

String DAC (D/A převodník s žebříčkovou sítí)

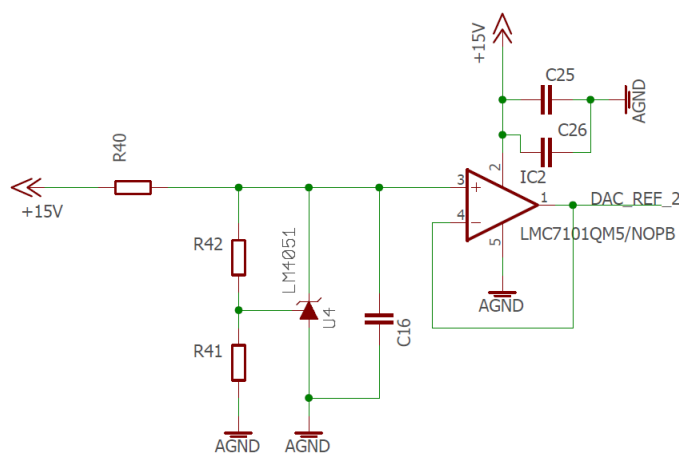
Jedná se o obecné označení D/A převodníku založeném na principu žebříčkové sítě tvořené rezistory. Přiváděné napětí V_{ref} je rozděleno pomocí napěťového děliče vytvořeném s pomocí rezistorů a jeho hodnota rozhoduje o maximálním výstupním napětí, které můžeme na analogovém výstupu očekávat. Pro převodník je zapotřebí 2^n rezistorů a spínačů, kde n je rozlišení převodníku a zároveň tak počet bitů binárního slova přiváděného na vstup. Pro 12bit. převodník je tedy zapotřebí až 4096 rezistorů spolu se spínači následujícími za těmito rezistory.

Výstupní analogová hodnota závisí na kombinaci bitů, které ovládají jednotlivé spínače a tak výstupní analogovou hodnotu. Pro konkrétní digitální kód dostáváme neměnnou analogovou hodnotu na výstupu. Kombinace vstupního digitálního slova se však neustále mění a tak se s každou jeho kombinací vytváří jedinečná analogová hodnota, jejichž konečný počet tvoří výsledný výstupní signál. Tento signál se poté podobá spojitě sinusové funkci, avšak o míře vyhlazení tohoto signálu na výstupu rozhoduje rozlišení použitého převodníku.

Rozlišení, jinými slovy, velikost kroku je dána vztahem $V_{ref}/2^n$, kde n je počet bitů převodníku. Pro 12bit převodník a $V_{ref} = 5\text{ V}$ je krok roven hodnotě 1,22 mV. Velikost kroku pojednává o nejmenší změně, ke které může dojít na analogovém výstupu v důsledku změny digitálního vstupu. Pokud tedy změníme stav LSB (nejméně významový bit), výstup se změní o 1,22 mV. [5]

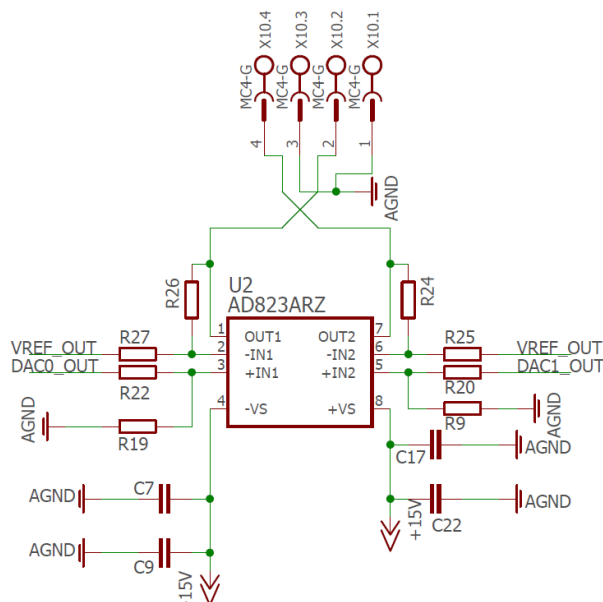
Vlastnosti D/A převodníků *DAC0* a *DAC1* vyvedených na desce

Náš mikrokontrolér obsahuje dva 12bit. D/A převodníky. Na vstup V_{DDA} D/A převodníku přivádíme napájecí napětí o hodnotě 3,3 V. Stabilizované napětí 3,3 V získáme obvodovým zapojením nacházejícím se na *obr. 13*. Z tohoto napětí je odvozena hodnota referenčního napětí přiváděného na vstup DAC (výběr zdroje referenčního zdroje napětí vybíráme softwarově pomocí nástroje *Processor Expert* - v našem případě se nyní jedná o externí zdroj, tedy ref. nap. = V_{DDA}). Tímto referenčním napětím jsme určili rozsah převodníku od 0 – 3,3 V (přivedením hodnoty 0x000h bude odpovídat výstupní analogová hodnota napětí 0 V, přivedením hodnoty 0xFFFFh získáme výstupní napětí kolem 3,3 V).



Obr. 13: Obvodové zapojení pro získání stabilizovaného napájecího napětí 3,3 V [3]

Abychom zvětšili rozsah našich D/A převodníků, je na desce umístěn (viz. obr. 14) duální operační zesilovač, který nám zajistí zvětšení rozsahu z 0 až 3,3 V na -10 až 10 V. Na invertující vstupy obou operačních zesilovačů je přiváděno referenční napětí (VREF_OUT = 3,3 V) a na neinvertující vstupy jsou přiváděny výstupní napěťové hodnoty z D/A převodníků (DAC0/1_OUT). D/A převodníky budou ve výsledku bipolární.



Obr. 14: Obvod v zapojení s duálním operačním zesilovačem pro zvětšení rozsahu obou D/A převodníků [3]

Výpočet výstupního napětí D/A převodníku

Pro výpočet výstupního napětí D/A převodníku využijeme vztahu:

$$U_{vyst} = N \cdot \frac{S_{max} - S_{min}}{2^n} + S_{min} \quad 2.3.1$$

kde

Rozsah převodníku ($S_{max} - S_{min}$)... 20 V

Počet bitů převodníku (n) ... 12

12bitové slovo (N) ... dec. 0 – 4096

2.3.3 Postup řešení pro vytvoření funkce $\cos(x)$

Pro obě řešení se vždy omezíme na $\frac{1}{4}$ periody funkce. Budeme-li vědět řešení dané funkce v intervalu od 0 do $\pi/2$, jednoduše poté odvodíme výsledné hodnoty pro celou periodu.

a) Vytvoření pole hodnot y pro $\frac{1}{4}$ průběhu funkce

V tomto projektu deklarujeme pole o velikosti 257 prvků, jejichž hodnota se bude pohybovat od 0.0 do 1.0. Toto pole nám poskytne řešení $\frac{1}{4}$ periody funkce $\cos(x)$, tedy v intervalu od 0 do $\pi/2$.

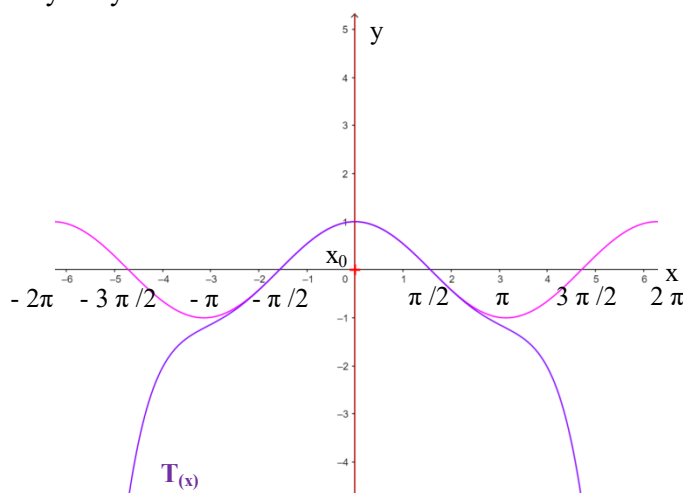
b) Taylorův polynom

Pomocí Taylorova polynomu jsme schopni aproximovat naši funkci, kterou nedovedeme spočítat, analogickou a jednodušší funkcí, jenž spočítat naopak umíme. [17]

Postup řešení $\sin(x)$ a $\cos(x)$ pomocí výpočtu *Taylorova polynomu*: [17]

$$T_n(x) = f(x_0) + \frac{f'(x_0)}{1!} \cdot (x - x_0)^1 + \frac{f''(x_0)}{2!} \cdot (x - x_0)^2 + \dots + \frac{f^n(x_0)}{n!} \cdot (x - x_0)^n \quad 2.5.1$$

Čím větší řád výpočtu n aplikujeme, tím přesnějších výsledků v širším intervalu x -vých hodnot dosahujeme. Pro naše účely nám bohatě postačuje výpočet polynomu 6. řádu, jehož aproximaci $T(x)$ naší funkce $\cos(x)$ vidíme na *obr. 15*. V intervalech od $-\pi/2$ do $\pi/2$ skutečně dosahujeme totožných výsledků.



Obr. 15: Aproximace funkce $\cos(x)$ v bodě $x_0 = 0$ pro $n = 6$ [4]

Pro výpočet funkce $\sin(x)$ můžeme rovněž vycházet z Taylorova polynomu 6. řádu pro funkci $\cos(x)$. Vzorce pro výpočty, pole pro $\sin(x)$ a řešení Taylorova polynomu se budou nacházet v nově založeném zdrojovém a hlavičkovém souboru pojmenovaném např. *Funkce.c* a *Funkce.h* v projektu.

2.3.4 Dosažené výsledky

Generované funkce $\sin(x)$ a $\cos(x)$ jsou zobrazeny na obr. 16. Funkce $\sin(x)$ s amplitudou 6 V je vytvářena s pomocí definované tabulky o velikosti $\frac{1}{4}$ její periody. Podprogram *f_{sin}_tab* této funkce hodnoty z tabulky dle příslušného algoritmu postupně vybírá. Může se zdát, že průběh funkce $\cos(x)$ o velikosti amplitudy cca 3 V je založen na stejném principu, jako již zmíněná $\sin(x)$, avšak tato funkce je nyní pro názorný příklad generovaná za pomoci Taylorova polynomu 6. řádu. Ať již tedy zvolíme jakýkoli postup, výsledek bude stejný nebo velmi podobný.



Obr. 16: Zobrazení generovaných funkcí sinus a cosinus na osciloskopu pro $f \approx 50$ Hz

Výběr dat z tabulky či výpočty dle Taylorova polynomu funkce $\sin(x)$ probíhají na principu periodické inkrementace proměnné *fw*, která nám tvoří pilovitý průběh definovaný o příslušné periodě, na jehož základě probíhá generování. Okamžité hodnoty obou funkcí jsou následně vypočteny dle známého vztahu $w(t) = A \cdot \sin(\omega t)$, kde A je požadovaná velikost amplitudy a $\sin(\omega t)$ funkce pro výběr hodnot z tabulky nebo výpočet Taylorova polynomu 6. řádu. Aby se takto definované hodnoty funkcí mohly zobrazovat a tvořit tak analogové průběhy, je nutné přivést je na D/A převodníky. Předtím je však nezbytné provést převod vypočtených okamžitých hodnot, během kterého se těmto hodnotám přiřadí potřebné napěťové hladiny a vytvoří se tak 16bitové číselné hodnoty, které se následně přivedou a zpracují pomocí D/A převodníků funkcí *SetValue16*. Použitá funkce *Recorder* nám vzorkuje hodnoty v rekordéru.

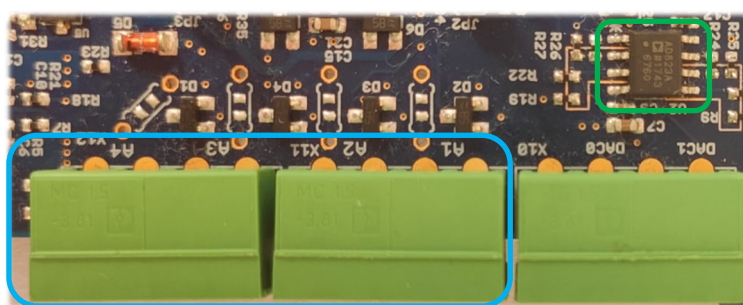
Příklad použití funkce *SetValue16*

```
void isr_pit0(void) { /* generovani funkce sinus */
    Uout_sin = Am_sin * fsin_tab(fwt); /* okamzita h. sinus => tabulka */
    Uout_sin_DAC0 = 204.7 * Uout_sin + 2047; /* 16bit. hodnota pro 1. DAC */
    DA0_SetValue16(&Uout_sin_DAC0); /* predani adresy 16bit. h. na 1. DAC */
    FMSTR1_Recorder(); /* funkce pro vzorkovani promennych v rekordéru */
}
```

2.4 Měření analogových signálů a jejich zpracování pomocí FreeMasteru

2.4.1 Popis úlohy

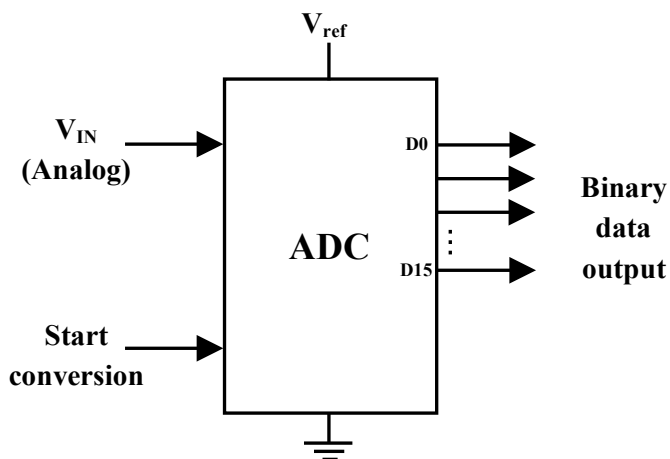
A/D převodníky jsou důležitou součástí pro měření a zpracovávání generovaných signálů. Jejich využitím se zabývá tato úloha, ve které se seznámíme s principem převodu generovaných analogových signálů přiváděných na vstupy převodníků do digitální podoby a vypočteme si hlavní parametry takto převedených signálů, jakými jsou střední a efektivní hodnota. Výpočet těchto parametrů budeme provádět pomocí grafické metody, která nám pomůže rozdělit periodu generovaného signálu na n stejných vzorků. Podstatnou část kapitoly bude tvořit podrobný popis zapojení a funkčnosti těchto převodníků vyvedených na vývojové desce. Na závěr se budeme věnovat konfiguraci příslušné komponenty v *Processor Expert* a řekneme si, které proměnné budeme chtít v jednotlivých oknech prostředí *FreeMasteru* zobrazovat.



Obr. 17: Umístění dvojice A/D převodníků (A1-A2, A3-A4) s vyznačeným duálním operačním zesilovačem AD823A pro zvětšení rozsahu na vývojové desce

2.4.2 A/D převodníky

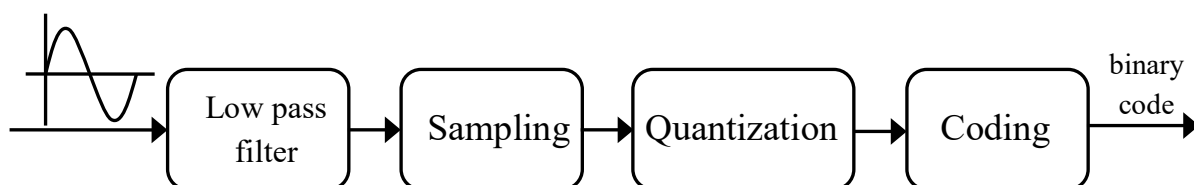
A/D převodníky jsou zařízení, která převádějí analogový signál spojitý v čase na diskrétní signál definovaný pouze v určitých časových okamžicích s následným převodem na číselnou hodnotu, se kterou je schopen mikropočítač dále pracovat.



Obr. 18: Blokové schéma A/D převodníku (podle [2])

Princip převodu zachycuje *obr. 19*. Na vstup převodníku se posílá analogový signál, který probíhá třemi fázemi převodu. Signál je nejdříve vzorkován pomocí vzorkovacího obvodu. Počet vzorků je dán rozlišením převodníku. V našem případě se jedná o 16bit. převodník, který poskytuje $2^{16} = 65536$ kroků. Velikost kroku, se kterým se bude analogový signál vzorkovat, závisí nejen na max. počtu kroků, ale také na velikosti přiloženého referenčního napětí. Pro ref. napětí 5 V u 16bit. převodníku je velikosti kroku dána vztahem $V_{ref}/2^{16}$. Výsledný krok je tedy roven hodnotě 0,076 mV a poskytuje nám informaci o nejmenší možné změně napětí na vstupu, kterou je schopen A/D převodník daný svým rozlišením rozeznat. Čím větší je rozlišení, tím menší je krok. Z toho vyplývá, že velikost kroku můžeme měnit velikostí přiváděného referenčního napětí V_{ref} . [2]

Vzorkovací frekvence by dle Nyquistova teorému měla dosahovat dvojnásobku maximální frekvence vstupního signálu, aby nedošlo k *aliasing* efektu, který v případě nedodržení této podmínky při zpětné rekonstrukci tohoto signálu způsobuje, že frekvence tohoto signálu bude odlišná oproti původní frekvenci vstupního signálu před vzorkováním. V případě vzorkování obdélníkového signálu k tomuto efektu dochází vždy, nehledě na velikosti vzorkovací frekvence, protože obdélníkový signál kromě základní harmonické obsahuje také vyšší harmonické složky signálu. Aby k tomuto efektu nedocházelo, zařazuje se k převodníku vyhlazovací filtr typu dolní propust (DP), který nám tyto vyšší harmonické odstraní a tak omezí účinek *aliasing* efektu. [18]

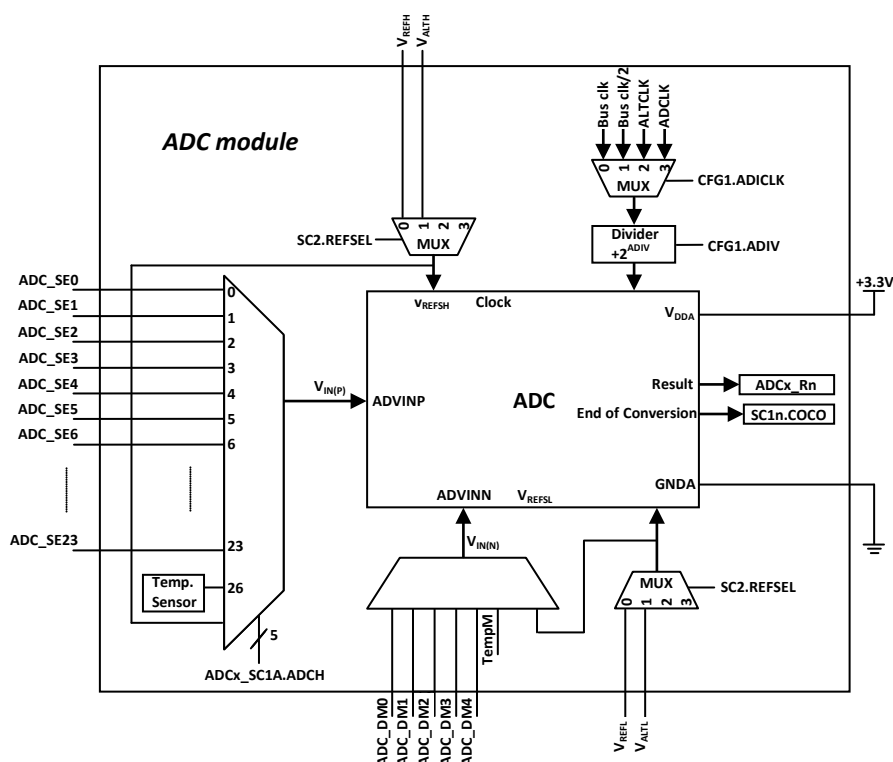


Obr. 19: Převod analogového signálu na digitální (diskrétní) (podle [5])

Následuje kvantování, při kterém jsou jednotlivé diskrétní vzorky zaokrouhleny na hodnoty, které odpovídají nejbližším kvantizačním hladinám. Počet těchto hladin je dán vztahem 2^n , kde n představuje rozlišení převodníku. Konečnou fází je kódování, při kterém se těmito kvantovacími hodnotám přiřazují odpovídající binární čísla, tedy posloupnost jedniček a nul, které mikropočítač již dokáže zpracovat.

Popis modulu ADC SAR (*Successive approximation – s postupnou aproximací*) [14] [2]

Použitý mikropočítač obsahuje dvojici A/D převodníků s rozlišením 16 bitů. Jeden modul ADC podporuje až 24 standardních jednostranných kanálů a také 4 diferenciální kanály. Na vývojové desce jsou vyvedeny pouze 4 jednostranné kanály každého z převodníků.



Obr. 20: Blokové schéma ADC modulu (podle [2])

Pro použití zvoleného převodníku je příslušnému modulu potřeba povolit a následně zvolit hodinový signál. Jsou k dispozici 4 možnosti zdrojů tohoto signálu, který nabízí konfigurační registr `ADCx_CFG1` (*Configuration Register 1*). Můžeme si vybrat úplný hodinový signál sběrnice *Bus clock*, který je mj. po resetu nastaven jako výchozí, nebo pouze jeho polovinu. Také můžeme vybírat z alternativního zdroje (`ALTCLK`) či asynchronních hodin (`ADCCLK`). `ADICLK` jsou generovány ze zdroje hodinového signálu nacházejícího se uvnitř ADC modulu. Výběr se provádí pomocí multiplexeru, jehož stav řídí bitové pole `ADICLK`. Na výstupu multiplexeru se nachází bitové pole `ADIV` poskytující děličku 2^{ADIV} vybraného hodinového signálu. Konfigurační registr také obsahuje bitové pole `MODE`, pomocí kterého lze vybrat rozlišení převodu mezi hodnotami 8, 12, 10 a 16bity.

Pomocí dalšího registru ADCx_SC1A, jehož bitové pole ADCH řídí multiplexer, se vybírá příslušný kanál převodníku k následnému převodu. Tento registr má pro výběr kanálu vyhrazeno 5 svých nejnižších bitů, které přísluší již zmíněnému bitovému poli. Tento registr mj. poskytuje diferenciální režim (DIFF), tedy možnost převodu rozdílu napětí mezi dvěma vstupy na diferenciálním analog. vstupu, a příznak dokončení konverze (COCO). Po skončení konverze můžeme její výslednou binární hodnotu číst z registru výsledků ADCx_Rn, ve kterém je po konverzi umístěna. Příznakový bit COCO registru ADCx_SC1n je po přečtení dat z ADCx_Rn automaticky vynulován.

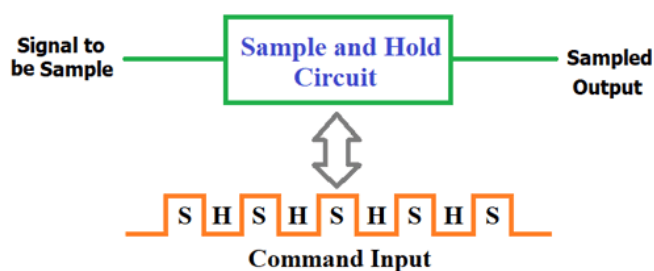
V_{REF} je vstupní napětí používané pro referenční napětí. A/D převodník lze nakonfigurovat tak, aby přijímal jedno ze dvou napětí externích referenčních párů (V_{REFH} , V_{REFL}). Externí ref. napětí V_{REFH} je připojeno na napájecí napětí V_{DDA} mikropočítače s hodnotou +3,3V. V_{REFL} je naopak spojeno se zemí V_{SSA} . Piny V_{ALTH} a V_{ATL} slouží jako alternativní a jsou založené na konfiguraci mikropočítače. Tyto referenční hodnoty napětí se vybírají pomocí registru SC2 pole REFSEL, který řídí stav multiplexeru. V případě potřeby diferenciálního referenčního napětí se využívá vztahu $V_{REF} = +V_{REF} - (-V_{REF})$, kde $+V_{REF}$ slouží pro nastavení samotné referenční hodnoty V_{REF} a $-V_{REF}$ je připojen k zemi.

Čas převodu (*Conversion time*)

Tento parametr nám určuje, za jakou dobu zvolený A/D převodník převede vstupní analogový signál na digitální výstup ve formě binárních hodnot. Čas je daný zvoleným zdrojem hodinových impulsů a technologií použitou při výrobě tohoto převodníku.

Vzorkovací obvod [19] [6]

A/D převodníky, jak již bylo zmíněno, tvoří vzorkovací obvod. Jedná se o analogové zařízení pracující na principu vzorkovacího a přidržovacího obvodu (*Sample and hold*), které vstupní analogový signál, tedy napětí, vzorkuje a následně ho po nastavenou dobu udržuje na konstantní hodnotě. Tento systém eliminuje případné změny narušující proces převodu. Tento obvod většinou tvoří kondenzátor udržující vzorkovaný vstupní signál, FET tranzistor jako spínací prvek a operační zesilovač. Následující blokové schéma popisuje princip činnosti tohoto obvodu.

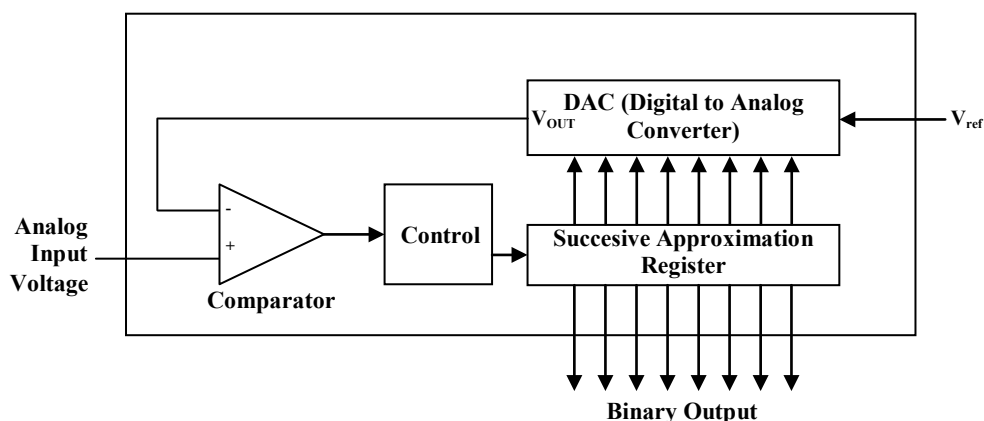


Obr. 21: Blokový diagram obvodu *Sample and hold* [6]

Příkazový vstup (*Command Input*) slouží ke spuštění a zastavení vzorkování vstupního signálu pomocí příkazů řízených principem pulzní šířkové modulace (PWM). Příkazy jsou posílány na spínací obvod, který je v případě log. 1 (příkaz *sample*) sepnutý a obvod vstupní analogový signál vzorkuje. Během vzorkování se také nabíjí připojený kondenzátor. V případě log. 0, tedy příkazu *hold*, se tento spínač rozeptne a kondenzátor v sobě drží poslední navzorkovanou hodnotu po dobu potřebnou pro převedení této hodnoty na digitální. Tato hodnota je následně převedena na binární podobu pomocí metody postupné aproximace.

Metoda postupné aproximace

Pomocí 16bit. A/D převodníku získáváme 16bit. digitální data na výstupu. Následující schéma nám popisuje princip převodu metodou postupné aproximace, která je velmi široce používána.



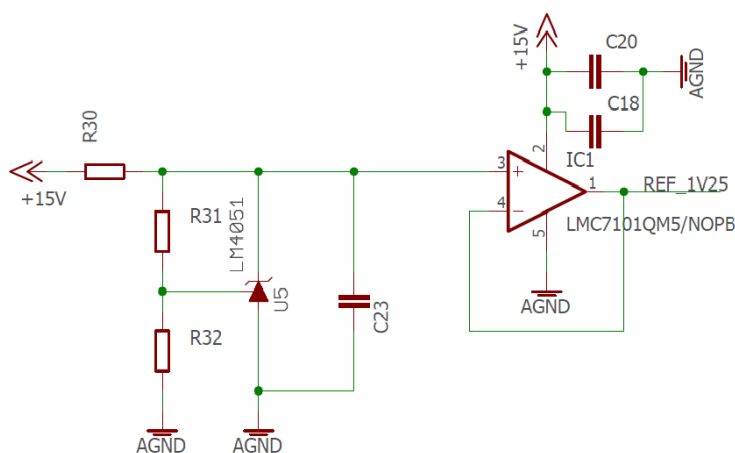
Obr. 22: Převod pomocí metody postupné aproximace (podle [2])

Hlavní části tohoto obvodu tvoří komparátor, aproximační registr (*SAR*) a kontrolní jednotka (*Control*). Aproximační registr je na počátku převodu, v případě 8bit. převodníku, načten na binární hodnotu 10000000. D/A převodník nám převádí binární hodnotu z registru *SAR* na analogovou hodnotu, která se spolu se vstupní hodnotou porovnává v komparátoru. Pokud předpokládáme velikost kroku 5 mV, pak analogová hodnota vypočtená z bin. hodnoty 10000000 a velikosti kroku je 128 (v dec.) $\times 5 \text{ mV} = 0,64 \text{ V}$. Následně dochází k porovnávání. Pokud je tato hodnota větší, než vstupní, pak se tato hodnota nastaví na log. 0. V případě, že je hodnota menší, než vstupní, log. 1 zůstane. Následuje bin. číslo 11000000, přičemž postup uvedený výše zůstává stejný, až nakonec dojde k 8 bitům. [2]

Vlastnosti A/D převodníků vyvedených na desce

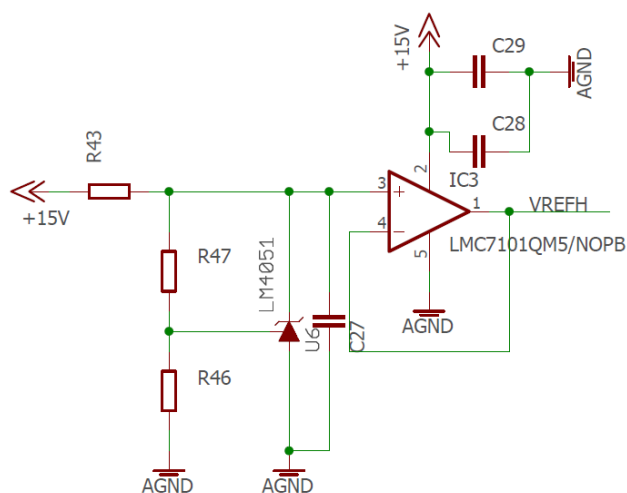
Náš mikrokontrolér obsahuje dva vícekanálové 16bitové A/D převodníky. Ke každému převodníku máme k dispozici 4 kanály, z nichž 2 kanály jsou určeny pro měření v rozsahu od -5 V do 5 V a další 2 kanály jsou pro měření od V_{REFL} do V_{REFH} . Kanály jsou multiplexerem přepínány ke společnému A/D převodníku.

Následující obvodové zapojení na *obr. 23* slouží pro nastavení stabilizovaného referenčního napětí pro neinvertující vstup každého rozdílového zesilovače určeného pro každou dvojici kanálů A/D převodníků. Pro přesné nastavení referenčního napětí nám slouží *LM4051*, jehož odpory *R30* až *R32* slouží pro nastavení tohoto napětí. Obvod také tvoří impedanční převodník, na jehož výstupu dostáváme referenční napětí nezávislé na zatížení.



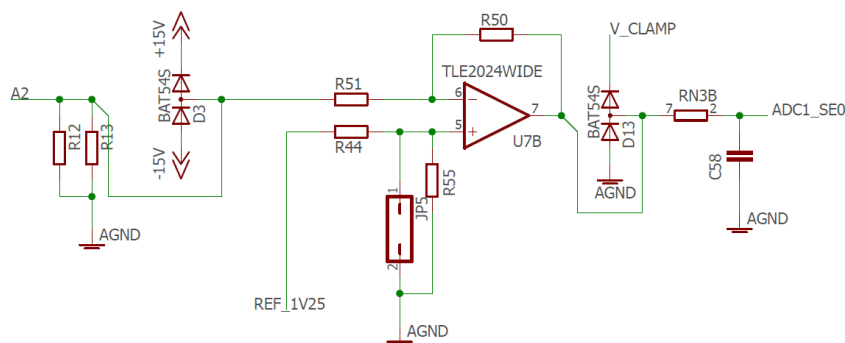
Obr. 23: Obvodové zapojení pro získání stabilizovaného referenčního napětí 1,25 V pro rozdílový zesilovač [3]

Další obvodové zapojení na *obr. 24* slouží pro nastavení stabilizovaného referenčního napětí V_{REFH} pro oba A/D převodníky.



Obr. 24: Obvodové zapojení pro získání stabilizovaného referenčního napětí úrovně „H“ pro A/D převodníky [3]

Následuje obvodové zapojení s OZ pro zvětšení měřicího rozsahu od -5 V do 5 V. Rozdílový zesilovač přivedené napětí na svorku A2, tedy na invertující vstup, upraví pro dovolený rozsah převodníku. Sériové zapojení diod (Clamp diode) slouží k ochraně OZ a A/D převodníku proti zvýšenému napětí překračujícímu hraniční úroveň.



Obr. 25: Obvodové zapojení s operačním zesilovačem pro zvětšení rozsahu A/D převodníku [3]

Výpočet číselné napět'ové hodnoty z n-bitového slova z A/D převodníku

Při A/D převodu získáváme 16bit. hodnotu. Abychom s touto hodnotou mohli dále pracovat, musíme ji převést na číselnou napět'ovou hodnotu dle následujícího vztahu:

$$U_{vyst} = -N \cdot \frac{S_{max} - S_{min}}{2^n} + S_{max} \quad 2.4.1$$

kde

Rozsah převodníku ($S_{max} - S_{min}$)... 10 V

Počet bitů převodníku (n) ... 16

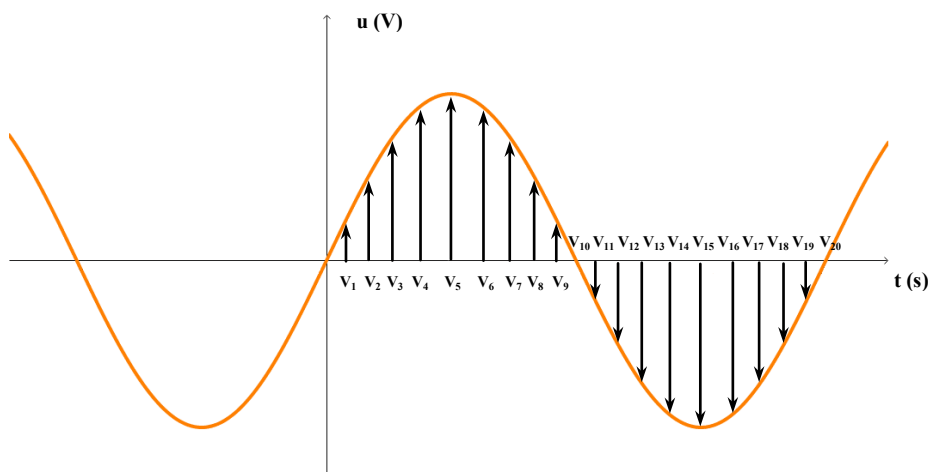
16bitové slovo (N) ... dec. 0 – 65535

2.4.3 Výpočet střední a efektivní hodnoty

V tomto projektu budeme počítat střední a efektivní hodnotu napětí.

Střední hodnota napětí [20]

Pro výpočet střední hodnoty napětí použijeme grafickou metodu:



Obr. 26: Rozdělení periody průběhu na stejný počet částí

Celou periodu našeho průběhu si rozdělíme na počet stejných částí n (400 vzorků). Každý i -tý vzorek napětí (V_i) nyní přičítáme k předešlým vzorkům a tak získáváme součet všech vzorků napětí celé periody. Tento součet podělíme počtem vzorků a získáme tak střední hodnotu periody průběhu.

$$V_{AVG} = \frac{V_1 + V_2 + V_3 + V_4 + V_5 \dots + V_{400}}{n} \quad 2.4.2$$

kde

V_1 až V_{400} ... velikost napětí jednotlivých vzorků

n ... počet vzorků

Součet všech vzorků budeme provádět dle následujícího vztahu:

$$V_{sum} = V_{sum} + V_i - V_{i-T} \quad 2.4.3$$

kde

V_{sum} ... součet všech vzorků

V_i ... vzorek aktuální periody

V_{i-T} ... i -tý vzorek předchozí periody

S pomocí tohoto vztahu (2.4.3) počítáme součet napětí všech vzorků v každém vzorkovacím čase. Vztah je doplněn o člen V_{i-T} , který nám určuje hodnotu napětí vzorku pořízeného v předchozí periodě. Bude-li se tedy průběh periodicky opakovat, bude platit $V_i = V_{i-T}$. Pomocí tohoto členu pak zastavíme přičítání, jelikož rozdíl $V_i - V_{i-T}$ bude nulový. Zvětšíme-li amplitudu průběhu, pak platí $V_i > V_{i-T}$. Dojde tak k přičtení rozdílu a tedy k aktualizaci V_{sum} na aktuální hodnotu součtu napětí všech vzorků. Zmenšíme-li amplitudu, pak platí $V_i < V_{i-T}$. Dojde tak k odečtení rozdílu a tedy opět k aktualizaci V_{sum} na aktuální hodnotu součtu napětí všech vzorků.

Výslednou střední hodnotu získáme následně:

$$V_{AVG} = \frac{V_{sum}}{n} \quad 2.4.4$$

kde

V_{AVG} ... výsledná střední hodnota napětí

V_{sum} ... součet všech vzorků

n ... počet vzorků

Efektivní hodnota napětí [21]

U efektivní hodnoty postupujeme stejným způsobem popsaným výše s rozdělení periody na počet stejných částí n . Následující vztahy se budou nepatrně lišit. Rozdíl bude v odmocnění výsledného součtu všech vzorků podělených jejich celkovým počtem.

Efektivní hodnotu napětí nyní vypočítáme pomocí vztahu:

$$V_{RMS} = \sqrt{\frac{V_1^2 + V_2^2 + V_3^2 + V_4^2 + V_5^2 \dots + V_{400}^2}{n}} \quad 2.4.5$$

kde

V_1 až V_{400} ... velikost napětí jednotlivých vzorků

n ... počet vzorků

Součet všech vzorků budeme provádět dle následujícího vztahu:

$$V_{sum} = V_{sum} + V_i^2 - V_{i-T}^2 \quad 2.4.6$$

kde

V_{sum} ... součet všech vzorků

V_i ... vzorek aktuální periody

V_{i-T} ... i -tý vzorek předchozí periody

Výslednou efektivní hodnotu získáme následně:

$$V_{RMS} = \sqrt{\frac{V_{sum}}{n}} \quad 2.4.7$$

kde

V_{RMS} ... výsledná efektivní hodnota napětí

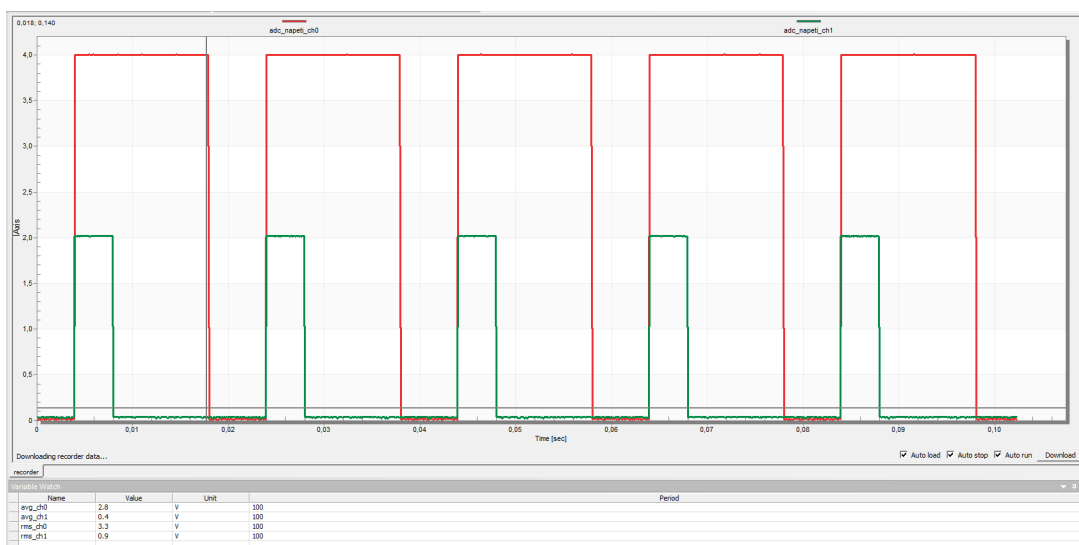
V_{sum} ... součet všech vzorků

n ... počet vzorků

Po vypočtení střední a efektivní hodnoty nesmíme zapomenout na uložení navzorkované i -té hodnoty do paměti (do pole o velikosti 400 prvků), kterou pak použijeme pro výpočet střední a efektivní hodnoty další periody signálu.

2.4.4 Dosažené výsledky

Vhodně nastavený generátor funkcí nám na dvoukanalový A/D převodník přivádí obdélníkové průběhy o určitých parametrech. Snímek pořízený ve *FreeMasteru* je možné vidět na *obr. 27*. A/D převodníky nám na svých kanálech zachycují a převádějí analogové hodnoty z generátoru do 16bitových digitálních hodnot s nastavenou periodou a ukládají do své paměti, ze které se pomocí funkcí komponent převodníků nabízených v *Processor Expert* vybírají, ukládají do pole, převádějí na číselné napěťové hodnoty a nakonec se pomocí příslušných vzorců zmíněných výše z těchto upravených hodnot tvoří generovaný periodický signál vypočítávají parametry, mezi které patří střední a efektivní hodnota tohoto signálu.



Obr. 27: Snímek generovaných obdélníkových průběhů ve *FreeMasteru* s parametry:
 adc_napeti_ch0 : $f = 50$ Hz, $Amp = 4$ Vpp, $Offset = 2$ V, $Phase = 0^\circ$, $Duty = 70\%$
 adc_napeti_ch1 : $f = 50$ Hz, $Amp = 2$ Vpp, $Offset = 1$ V, $Phase = 0^\circ$, $Duty = 20\%$

Příklad použití funkce *GetChanValue16* s výpočtem parametrů periodického signálu

```

void isr_pit0(void) {
    AD1_Measure(FALSE); /* zachycení a převod analog. hodnoty do 16bit. digit.
                          hodnoty s uložením do paměti ADC pro každý kanál */
}

void isr_adc(void) { /* Zobrazení a výpočet parametru gen. period. průběhu */
    AD1_GetChanValue16(ADC_chan_0, adc);
    /* vyber a uložení poslední namerené hodnoty kanálu 0 do prvku pole */

    adc_napeti_ch0 = -(adc[0] * 1.525878906e-4) + 5;
    /* převod 16bit. hodnoty na číselnou napět. hodnotu */

    /* Výpočet střední hodnoty */
    avg_ch0_soucet += adc_napeti_ch0 - pole_ch0[p];
    /* V_sum = V_sum + V_i - V_(i-T) */
    avg_ch0 = avg_ch0_soucet / 400; /* V_AVG = V_sum / n */

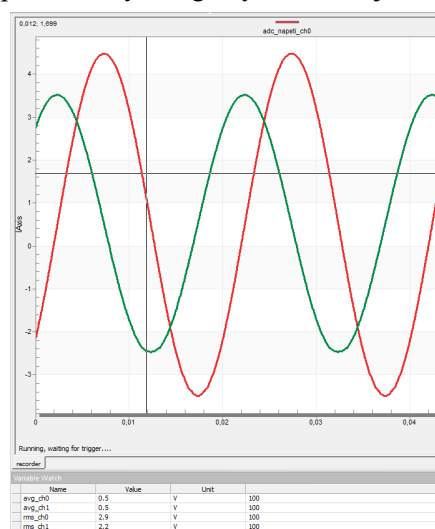
    /* Výpočet efektivní hodnoty */
    rms_ch0_soucet += adc_napeti_ch0*adc_napeti_ch0 - pole_ch0[p]*pole_ch0[p];
    /* V_sum = V_sum + (V_i)^2 - (V_(i-T))^2 */
    rms_ch0 = odmocnina(rms_ch0_soucet / 400); /* V_RMS = sqrt(V_sum / n) */

    p++; /* inkrementace pro následující prvek pole */
    if(p == 400) { /* max. 400 vzorků/hodnot */
        p = 0; /* vynuluj */
    }

    FMSTR1_Recorder(); /* funkce pro vzorkování promenných v rekorderu */
}

```

Takto je možné generovat libovolné periodické signály o různých parametrech nastavených v generátoru. Obr. 28 znázorňuje generování funkcí $\sin(x)$ a $\cos(x)$, které jsou posílány na dvoukanálový A/D převodník. Tento převodník s určitou periodou tyto signály navzorkuje a vytvoří tak, v našem případě, 400 vzorků, ze kterých se vypočítají parametry signálu a přiváděné signály spolu s těmito parametry se zobrazí také ve *FreeMasteru*.

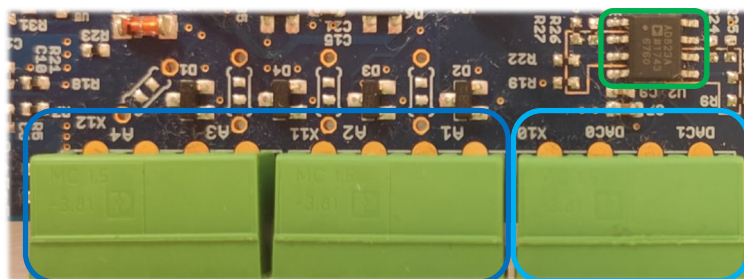


Obr. 28: Snímky generovaných funkcí z osciloskopu (vlevo) a ve *FreeMasteru* (vpravo)

2.5 Návrh a realizace struktury regulace napětí na RC členu pomocí PI regulátoru

2.5.1 Popis úlohy

Uživatel si na této úloze vyzkouší návrh PI regulátoru, pro který spočítá jeho dílčí parametry. Mezi tyto parametry charakterizující regulátor se řadí zesilovací konstanta K_R a časová konstanta T_R . Navržený regulátor bude regulovat výstupní napětí jednoduchého setrvačného členu 1. řádu v podobě RC článku, který bude připojený na jeho výstupu. V této úloze budeme využívat oba již zmíněné typy převodníků, z nichž vstupy A/D převodníku budou sloužit k přivedení požadované hodnoty z generátoru obdélníkového signálu a skutečné hodnoty z výstupu připojeného článku, zatímco D/A převodník nám poslouží k přivedení vypočtené hodnoty akčního zásahu na vstup RC článku. Úloha nás provede výpočtem parametrů PI regulátoru pomocí optimalizačních metod, mezi které patří metoda optimálního modulu (OM) a symetrického optima (SO). Průběhy nejdůležitějších veličin při regulaci dané soustavy, mezi které se řadí akční zásah a skutečná hodnota spolu s požadovanou, budeme pozorovat jak na osciloskopu, tak také na samotném *FreeMasteru*, s jehož pomocí si vyzkoušíme, jak se průběh těchto veličin změnou konstant regulátoru mění. Nesmíme zapomenout na změření odezvy navrženého PI regulátoru na vstupní pravoúhlý signál, která tvoří podstatnou a nepostradatelnou část naší úlohy.



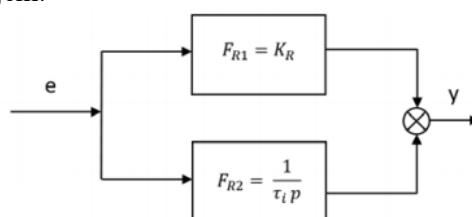
Obr. 29: Umístění A/D (A1-A2, A3-A4) a D/A (DAC0, DAC1) převodníků na vývojové desce

2.5.2 PI regulátor

Regulátory patří mezi nejdůležitější řídicí členy regulačních obvodů a naleznou patřičné využití při řízení elektrických pohonů pomocí mikropočítačů. Poskytují nám požadované dynamické vlastnosti regulovaných soustav. V tomto projektu se budeme zabývat regulátorem typu PI.

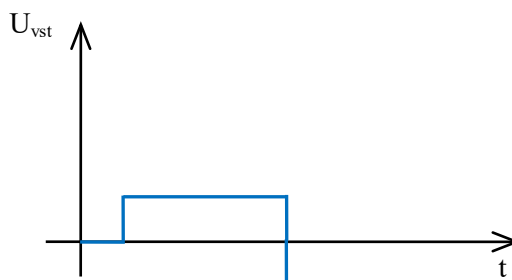
PI regulátor (proporcionálně integrační)

PI regulátor patří k nejpoužívanějším typům regulátorů v elektrických pohonech. Skládá se z P a I členu tvořící paralelní zapojení.

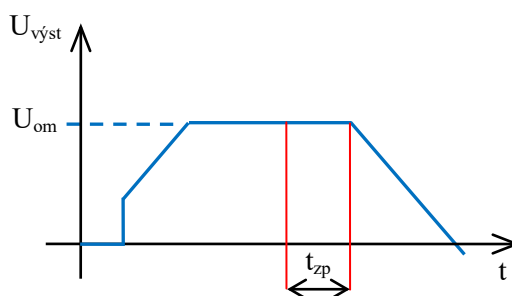


Obr. 30: Blokové zapojení složek PI regulátoru [7]

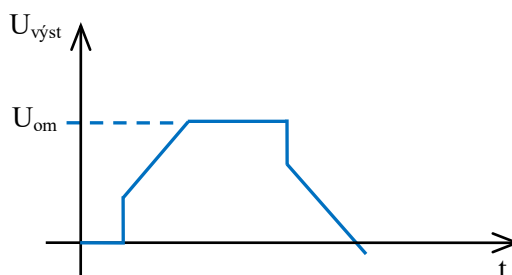
Přechodová charakteristika PI regulátoru nám poskytuje odezvu chování regulátoru na vstupní skokový signál. Reálný PI regulátor je vybaven vnějším a vnitřním omezovačem. Pro lepší představu je vliv těchto omezovačů zobrazen na následujících obrázcích (obr. 31 - 33).



Obr. 31: Zobrazení jednotkového skoku na vstupu PI regulátoru pro zjištění přechodové charakteristiky (podle [7])



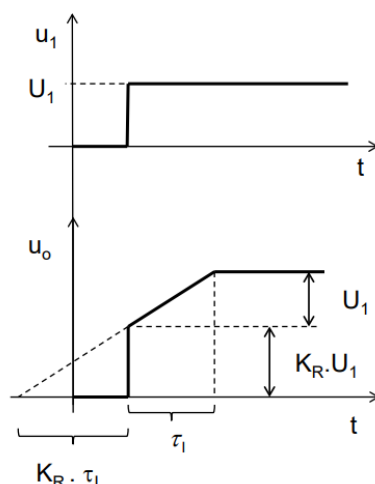
Obr. 32: Regulátor bez vnitřního omezovače s vyznačením zpoždění reakce t_{zp} (podle [7])



Obr. 33: Regulátor s vnitřním omezovačem (bez zpoždění, $t_{zp} = 0$) (podle [7])

Vnější omezení je nastavitelné a jeho maximální hodnota je dána použitím D/A převodníku s rozsahem ± 10 V (U_{om}). Výše můžeme vidět vliv vnějšího omezovače. Bude-li tedy výstupní napětí z regulátoru růst nad námi nastavenou hodnotu U_{om} dojde k omezení výstupní hodnoty. Nevýhodou tohoto řešení je zpomalení reakce regulátoru na změnu požadované hodnoty způsobeného vlivem integrační složky. Z tohoto důvodu je potřeba regulátor vybavit také vnitřním omezovačem, který zastaví integraci při dosažení omezení a tak zajistí výstupní signál bez zpoždění. Výsledkem je pak téměř okamžitá reakce regulátoru na změnu vstupní požadované hodnoty. [7]

Z přechodové charakteristiky je možné odečíst zesílení proporcionální složky (K_R) a časovou konstantu složky integrační (T_I) z níž je možné dopočítat nastavenou časovou konstantu regulátoru.

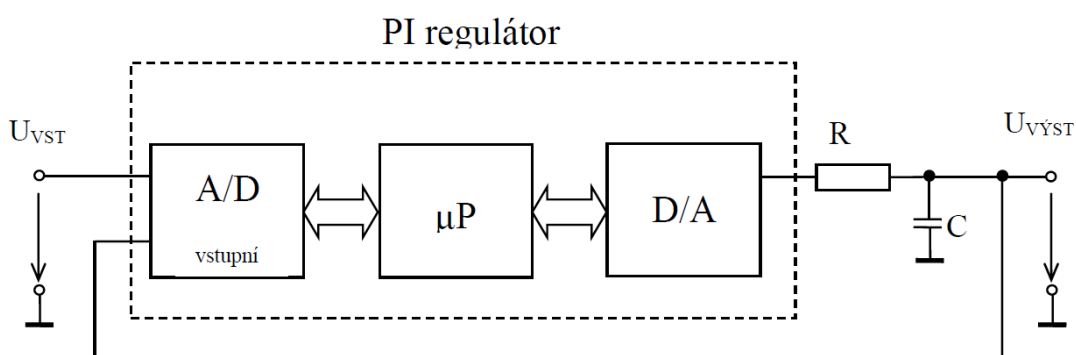


Obr. 34: Přechodová charakteristika PI regulátoru s odezvou na skokovou hodnotu napětí [7]

2.5.3 Návrh PI regulátoru [9]

PI regulátor se skládá ze čtyřkanálového A/D převodníku, z nichž pouze 2 kanály jsou použity k samotné regulaci (např. kanály $A1$, $A2$). Na jeho vstupy se přivádí požadovaná a skutečná hodnota napětí na kondenzátoru přes zpětnou vazbu. Naším úkolem je zajistit na výstupu průběh napětí, jehož tvar a velikost určuje právě přiváděné vstupní napětí (U_{vst}).

Následuje blok samotného mikropočítače (μP), který zpracovává regulační odchylku, kterou získává z rozdílu mezi požadovanou a skutečnou hodnotou napětí. Pomocí této regulační odchylky mikropočítač vypočítá regulační zásah, který vhodně přizpůsobí pro zpracování D/A převodníkem.

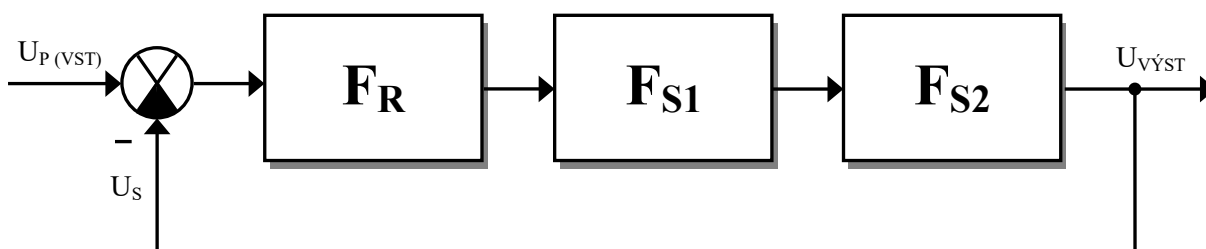


Obr. 35: Blokové zapojení PI regulátoru k RC filtru [8]

Regulovaným obvodem je v našem případě RC článek, tedy setrvačný článek 1. řádu se zesílením $K_s = 1$, který nám napodobuje reálné chování soustav, se kterými se ve většině případů v praxi setkáváme. Časová konstanta PI regulátoru je v tomto případě rovna čas. konstantě RC článku.

Návrh regulátoru provedeme pomocí optimalizačních metod/kritérií, mezi které se řadí optimální modul (OM) a symetrické optimum (SO). Návrh spočívá ve správném zvolení typu regulátoru (v našem případě PI), který bude kompenzovat největší časovou konstantu v obvodu. Tato optimalizační kritéria kromě parametrů určují také vhodný typ regulátoru.

Schéma regulačního obvodu:



Obr. 36: Blokové zapojení regulačního obvodu

kde:

U_P = požadovaná hodnota

U_S = skutečná hodnota

$F_{S1}(p)$... přenos RC článku

$F_{S2}(p)$... přenos A/D a D/A převodníku

1) Vyjádříme si přenos PI regulátoru $F_R(p)$ ze vztahu pro výpočet otevřené smyčky

$$F_0(p) = F_R(p) \cdot F_S(p) \Rightarrow F_R(p) = \frac{F_0(p)}{F_S(p)} \quad 2.5.1$$

kde

$F_0(p)$... přenos otevřené smyčky

$F_R(p)$... přenos regulátoru

$F_S(p)$... přenos soustavy ($F_{S1}(p)$ a $F_{S2}(p)$)

2) Vypočteme přenos soustavy

$$F_S(p) = F_{S1}(p) \cdot F_{S2}(p) = \frac{K_1}{1 + pT_1} \cdot \frac{K_2}{1 + p\tau_\sigma} = \frac{K_s}{(1 + pT_1) \cdot (1 + p\tau_\sigma)} \quad 2.5.2$$

kde

K_1 ... zesílení RC článku

K_2 ... zesílení převodníků

$K_s \dots$ zesilovací konstanta soustavy ($K_s = 1$)

$\tau_1 \dots$ největší časová konstanta v obvodu, kterou budeme kompenzovat
(v našem případě se bude jednat o čas. konstantu RC článku)

$\tau_\sigma \dots$ součet malých časových konstant soustavy (A/D a D/A převodníky)

3) Vypočteme časové konstanty obvodu

$$T_1 = R \cdot C \quad 2.5.3$$

$$\tau_\sigma = 2 \cdot T_{vz} \quad 2.5.4$$

kde

$T_1 \dots$ časový konstanta RC článku

$\tau_\sigma \dots$ součet malých časových konstant
(v našem případě se tedy jedná o součet vzorkovací periody D/A a A/D převodníku)

$T_{vz} \dots$ vzorkovací perioda

4) Dle přenosu otevřených smyček optimalizačních metod vypočítáme samotný přenos regulátoru

OM: (zpožďující člen 2. řádu)

$$F_{0(OM)} = \frac{1}{2p\tau_\sigma(1 + p\tau_\sigma)} \quad 2.5.5$$

SO: (astatismus 2. řádu)

$$F_{0(SO)} = \frac{1 + 4p\tau_\sigma}{8p^2\tau_\sigma^2(1 + p\tau_\sigma)} \quad 2.5.6$$

a) Výpočet přenosu regulátoru dle OM:

$$F_{R(OM)} = \frac{F_{0(OM)}}{F_s} = \frac{\frac{1}{2p\tau_\sigma(1 + p\tau_\sigma)}}{\frac{K_s}{(1 + pT_1) \cdot (1 + p\tau_\sigma)}} = \frac{(1 + pT_1)}{2p\tau_\sigma K_s} = >$$

Nyní se snažíme docílit přiblížení tohoto přenosu k přenosu PI regulátoru:

$$F_{R(PI)} = K_R \cdot \frac{1 + pT_R}{pT_R} \quad 2.5.7$$

Proto provedeme následující úpravy:

$$\Rightarrow F_{R(OM)} = \frac{(1 + pT_1)}{2p\tau_\sigma K_s} \cdot \frac{pT_1}{pT_1} = \underbrace{\frac{T_1}{2\tau_\sigma K_s}}_{K_R} \cdot \underbrace{\frac{(1 + pT_1)}{pT_1}}_{T_R}$$

kde

$$\frac{T_1}{2\tau_\sigma K_s} = K_R$$

$$pT_1 = T_R$$

K_R ... zesilovací konstanta PI regulátoru

T_R ... časová konstanta PI regulátoru

b) Výpočet přenosu regulátoru dle SO:

Metodu symetrického optima můžeme použít pouze v případě astatických soustav (obsahující integrační člen). Budeme-li mít soustavu typu:

$$F_S(p) = \frac{K_s}{(1 + pT_1) \cdot (1 + p\tau_\sigma)} \quad 2.5.8$$

kde $T_1 \gg 4 \cdot \tau_\sigma$ (v našem případě tomu tak je)

Pak můžeme aproximovat setrvačný člen 1. řádu s největší časovou konstantou v obvodu (zpožďující člen 1. řádu) členem integračním s přenosem:

$$F_I(p) = \frac{1}{pT} \quad 2.5.9$$

a tak z této soustavy udělat soustavu astatickou.

V případě, že toto zjednodušení soustavy neprovedeme, dostaneme se k náročnějšímu výpočtu a návrhu PI^2D regulátoru.

Výsledný přenos soustavy po aproximaci bude nabývat tvaru:

$$F_S(p) = \frac{K_s}{pT_1 \cdot (1 + p\tau_\sigma)} \quad 2.5.10$$

Nyní vypočteme přenos regulátoru dle metody SO:

$$F_{R(S0)} = \frac{F_{0(S0)}}{F_S} = \frac{\frac{1 + 4p\tau_\sigma}{8p^2\tau_\sigma^2(1+p\tau_\sigma)}}{\frac{K_s}{(pT_1) \cdot (1+p\tau_\sigma)}} = \frac{(1 + 4p\tau_\sigma) \cdot pT_1}{8p^2\tau_\sigma^2K_s} \Rightarrow$$

V tomto případě jsme se již k přenosu PI regulátoru přiblížili, proto přenos pouze patřičně upravíme:

$$\Rightarrow F_{R(S0)} = \underbrace{\frac{T_1}{2K_s\tau_\sigma}}_{K_R} \cdot \underbrace{\frac{1 + 4p\tau_\sigma}{4p\tau_\sigma}}_{T_R}$$

kde

$$\frac{T_1}{2\tau_\sigma K_s} = K_R$$

$$4p\tau_\sigma = T_R$$

K_R ... zesilovací konstanta PI regulátoru

T_R ... časová konstanta PI regulátoru

c) Porovnání vztahů pro výpočet konstant PI regulátoru pomocí OM a SO

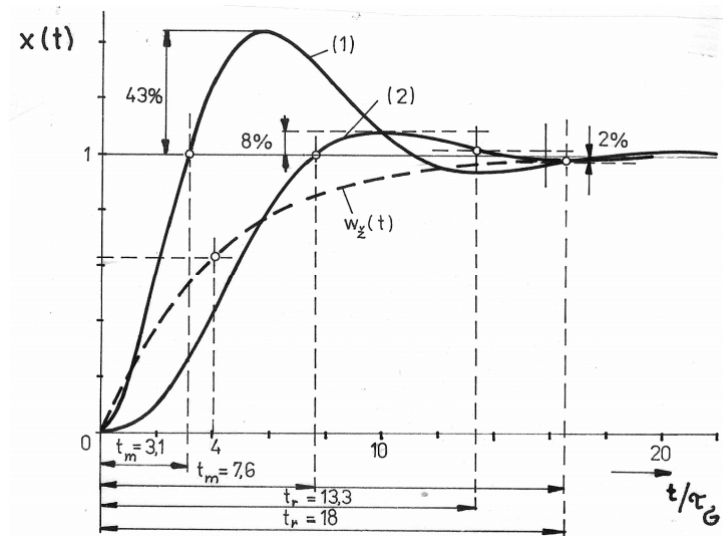
Z výše uvedených vztahů můžeme vidět, že pro výpočet zesilovací konstanty K_R PI regulátoru u obou metod vycházejí shodné vztahy, avšak u časových konstant je tomu jinak.

Regulátor vypočtený dle metody OM má obvykle větší časovou konstantu T_R oproti čas. konstantě vypočtené dle SO a dosahuje pomalejší rychlosti regulace (t_m). Avšak jeho výhodou je znatelně menší překmit σ , který dosahuje 4%, zatímco u SO je tomu 43%. Ke snížení tak velkého překmitu u metody SO se regul. obvod doplňuje filtrem, který tento překmit dokáže značně omezit a ještě k tomu zmenšit dobu regulace (t_r). V následující tabulce *tab. 2* je uvedeno srovnání parametrů přechodových charakteristik u jednotlivých optimalizačních metod OM a SO. [9]

Tab. 2: Srovnání parametrů přechodových charakteristik dle použitých optimalizačních metod

	OM	SO
τ_m	$4,7\tau_\sigma$	$3,1\tau_\sigma$
τ_r	$11\tau_\sigma$	$18\tau_\sigma$
σ	4 %	43 %

Na následujícím obrázku *obr. 37* můžeme vidět přechodové charakteristiky dle metody SO, na které je možné pozorovat vliv ukazatelů kvality regulace, mezi které patří rychlost regulace (t_m), doba regulace (t_r) a překmit (σ), bez použití filtru (1) a s filtrem (2), jež způsobí výrazné snížení překmitu z 43% až na 8% a podstatně sníží dobu regulace.



Obr. 37: Přechodové charakteristiky dle metody SO bez a s vlivem filtru s vyznačením charakteristických parametrů [9]

2.5.4 Realizace PI regulátoru [8]

Následující rovnice popisuje chování ideálního PI regulátor bez omezení:

$$ah(t) = K_R \cdot e(t) + \frac{1}{T_I} \cdot \int e(t) \cdot dt \quad 2.5.11$$

kde

$ah(t)$... akční hodnota (výstup z regulátoru)

K_R ... konstanta zesílení regulátoru

T_I ... integrační časová konstanta regulátoru

$e(t)$... regulační odchylka (vstup regulátoru)

Při realizaci regulátoru pomocí mikropočítače jsou základní rovnice popisující chování regulátoru nejprve převedeny do číslicové podoby. Následně jsou takto převedené rovnice zavedeny do řídicích struktur s mikropočítačem.

Vypočtené parametry regulátoru dosazujeme do následujících vztahů:

Rovnice pro implementaci PI regulátoru bez omezení:

$$ah(k) = (K_2 \cdot e(k) + sum(k - 1))/A \quad 2.5.12$$

$$sum(k) = sum(k - 1) + K_1 \cdot e(k) \quad 2.5.13$$

kde

$ah(t)$... akční hodnota (výstup z regulátoru)

$sum(k)$... suma pro výpočet integrálu

$sum(k - 1)$... předchozí vypočtená hodnota $sum(k)$

K_1 ... přepočtená časová konstanta regulátoru

K_2 ... přepočtená zesilovací konstanta regulátoru

$e(k)$... regulační odchylka (vstup regulátoru)

kde:

$$K_1 = A \cdot \frac{K_R \cdot T_{vz}}{T_R} \quad 2.5.14$$

$$K_2 = K_R \cdot A \quad 2.5.15$$

A ... konstanta pro normování výpočtu (v našem případě $A = 1$)

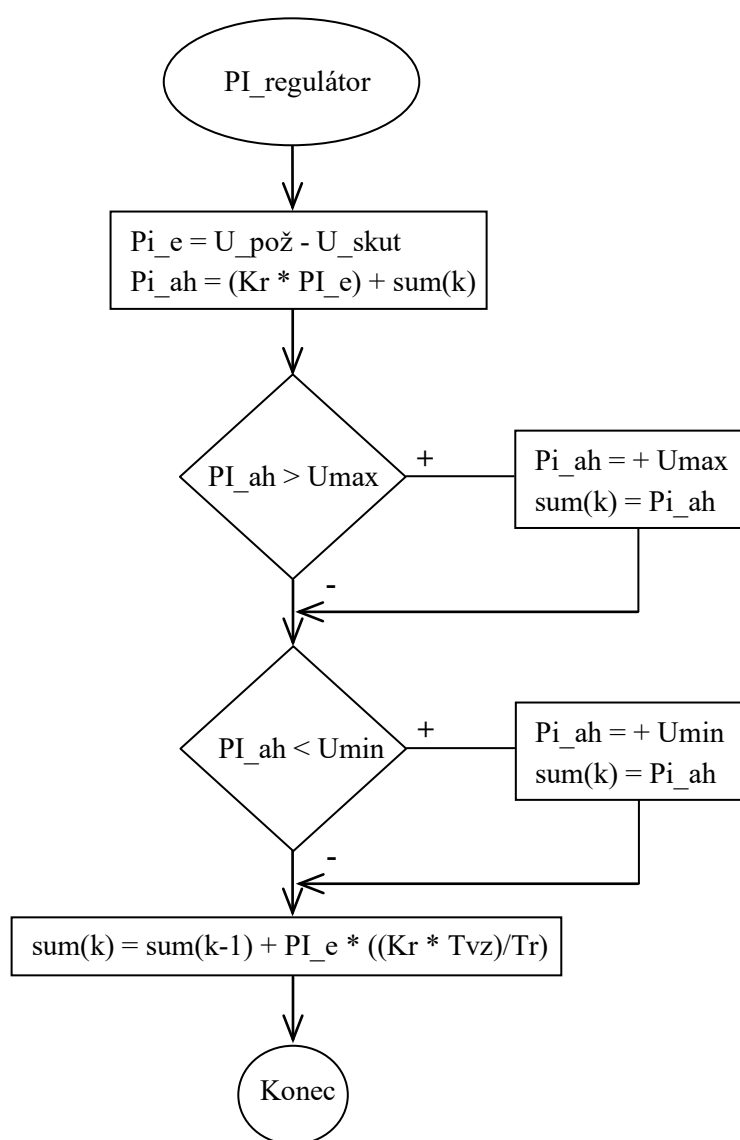
K_R ... konstanta zesílení regulátoru

T_{vz} ... vzorkovací perioda

$T_R = T_1$... časová konstanta RC článku = PI regulátoru

Vývojový diagram PI regulátoru

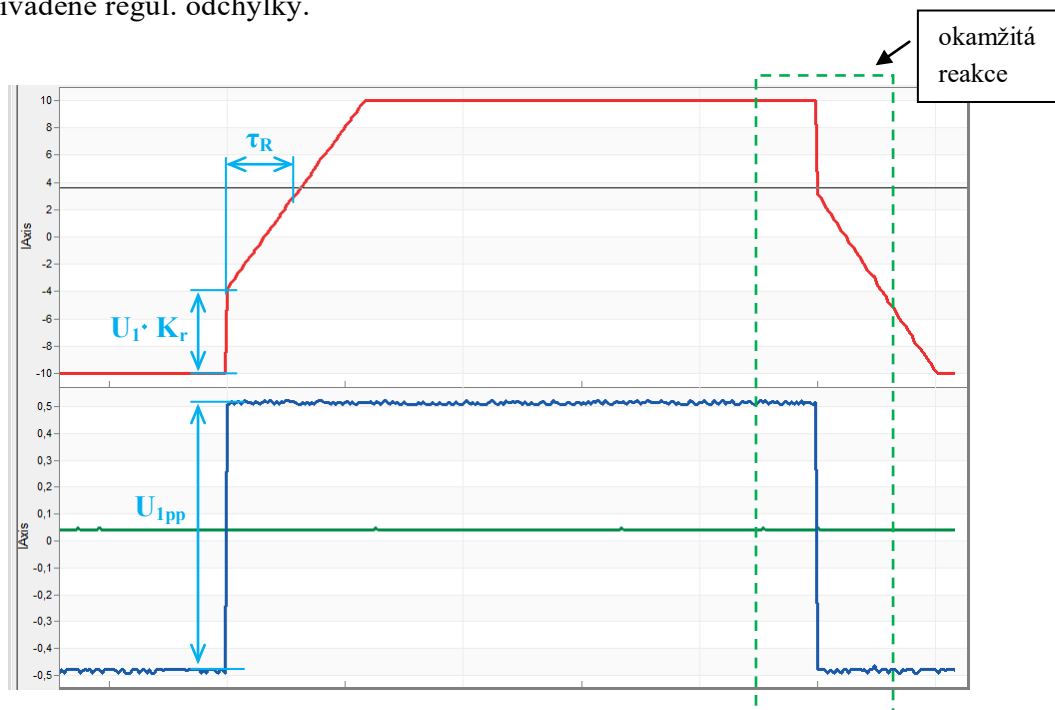
Po snazší představu principu fungování PI regulátoru založeného na mikropočítači je zde uveden vývojový diagram popisující všechny jeho důležité dílčí části.



Obr. 38: Vývojový diagram PI regulátoru realizovaný pomocí funkcí v číslicové podobě (podle [10])

2.5.5 Dosažené výsledky

Odezva PI regulátoru na pravoúhlý vstupní signál s vyznačením velikostí jednotlivých parametrů je zobrazena na *obr. 39*. Regulační odchylka přiváděná na vstup PI regulátoru o velikosti 0,5 V je zesílena o P složku a současně integrována I členem regulátoru. Na tomto obrázku také můžeme vidět vliv vnitřního omezovače, díky kterému je integrace omezena na cca +10 V, a okamžitou reakci regulátoru na změnu velikosti přiváděné regul. odchylky, kdy vnější omezovač nám omezí velikost výstupního napětí na požadovanou mez a díky vnitřnímu omezovači dojde k úplnému zastavení integrace, což umožní regulátoru okamžitě zareagovat na změnu přiváděné regul. odchylky.

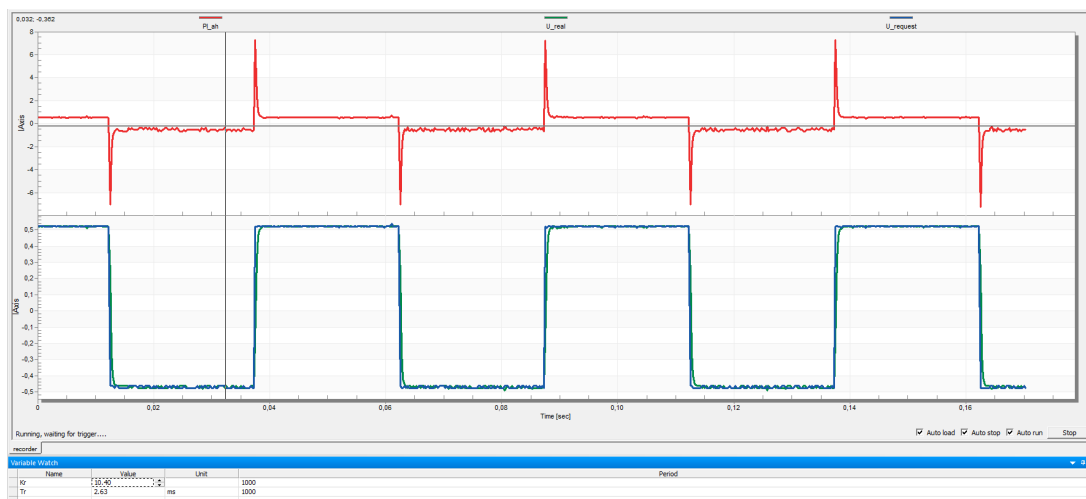


Obr. 39: Snímek odezvy navrženého PI regulátoru na vstupní pravoúhlý signál ve FreeMasteru s vyznačením parametrů a okamžité reakce/změny,
 $K_R = 13,15$, $T_R = 2,63 \text{ ms}$

Pro daný RC článek s parametry $R = 5,6 \text{ k}\Omega$ a $C = 470 \text{ nF}$ byl pomocí metody optimálního modulu (OM) navržen PI regulátor s časovou konstantou $T_R = 2,63 \text{ ms}$ a konstantou zesílení $K_R = 13,15$.

Regulační obvod tvoří 2kanálový A/D převodník pro přivádění skutečné a požadované hodnoty napětí a D/A převodník, který má za úkol přivést akční zásah v analogové podobě na vstup RC článku k regulaci jeho výstupního napětí.

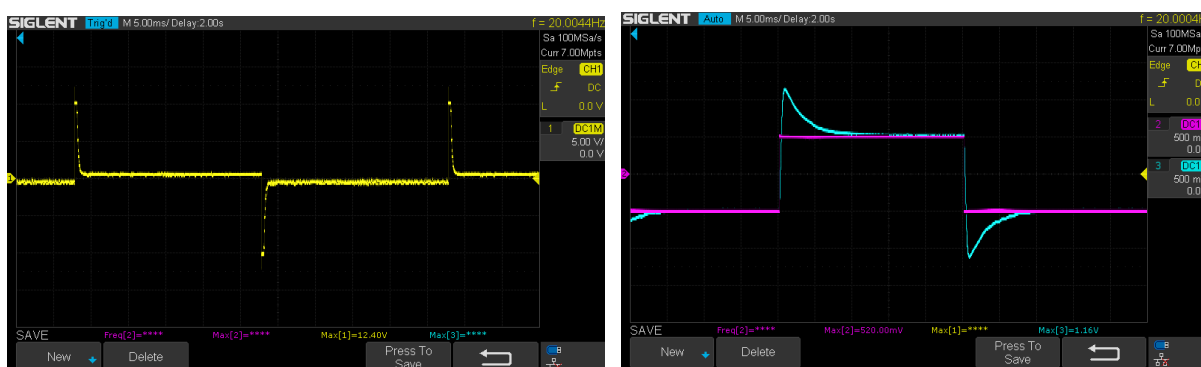
Na následujícím obrázku *obr. 40* je možné spatřit samotnou regulaci výstupního napětí regulované soustavy v podobě setrvačného článku 1. řádu, kdy vypočtená konstanta zesílení navrženého regulátoru byla pozměněna na hodnotu o velikosti 10,4, tedy takovým způsobem, aby se skutečná hodnota napětí na výstupu článku nejvíce blížila hodnotě požadované.



Obr. 40: Snímek regulace RC članku připojeného na výstup PI regulátoru ve FreeMasteru se změnou hodnoty konstanty zesílení

$$K_R = 10,4, T_R = 2,63 \text{ ms}, V_{pp} = 1 \text{ V}, f = 20 \text{ Hz}$$

Snímky probíhající regulace, tedy nejdůležitějších veličin při regulaci, pořízené z osciloskopu jsou na následujících snímcích. Při této regulaci se nastavily vypočtené parametry konstant PI regulátoru. Ze snímku požadované a skutečné hodnoty můžeme zpozorovat míru kvality regulace, která závisí na době a rychlosti regulace, tedy na parametrech určujících, za jaký čas a s jakou mírou stability se bude skutečný průběh napětí na výstupu RC članku shodovat s napětím požadovaným. Překmit, který se na tomto obrázku přechodných stavech nachází, je způsoben vysokým ziskem (K_R) proporcionální složky. Když se jeho zisk zvýší, pak regulátor sice reaguje na změny nastavené požadované hodnoty rychleji, avšak postupně se stává nestabilní.



Obr. 41: Snímky průběhu akčního zásahu (vlevo) a požadované a skutečné hodnoty napětí (vpravo) při regulaci RC članku připojeného na výstup PI regulátoru zaznamenaného z osciloskopu,

$$K_R = 13,15, T_R = 2,63 \text{ ms}, V_{pp} = 1 \text{ V}, f = 20 \text{ Hz}$$

2.6 Programové bloky pro řízení pohonů pomocí FreeMasteru

2.6.1 Popis úlohy

Úloha nás seznámí s různými programovými bloky určených pro vektorové řízení střídavých pohonů, pomocí nichž budeme požadovaný prostorový vektor, který nám může představovat napětí přiváděné na vstup střídavého stroje, převádět do různých komplexních souřadnicových systémů. Tyto převody mají svá opodstatnění při návrhu řídicího systému, které se dozvíme v následující kapitole. Opět si vyzkoušíme generování třífázového signálu, které je možné provádět odlišnými způsoby, převod tohoto signálu na dvoufázový a následně jednotlivé vektorové rotace, při nichž budeme pracovat s požadovanými úhly, které budou zadané, nebo vypočtené s pomocí vektorové analýzy. Příkladem může být vektor proudu, který můžeme natočit do rotorového souřadnicového systému v případě znalosti polohy rotoru.

2.6.2 Vektorové řízení [11, 22]

Pro regulaci točivého momentu jakéhokoli elektrického stroje je zapotřebí řídit velikost proudu. Na střídavý stroj, obvykle na trojfázové vinutí statoru, přivádíme trojfázový proud, jehož fáze jsou vzájemně pootočené o 120° . Tyto fázové proudy nám v souřadnicovém systému tvoří prostorový vektor statoru o určité velikosti a úhlu. Řízením těchto proudů je možné vytvořit vektor s požadovanými parametry.

Složky 3fázového statorového proudu tvořící prostorový vektor se napřed pomocí Clarkovy transformace $3/2$ převedou do neotáčivého statorového souřadného systému $\alpha\beta$ a tvoří tak příčnou a podélnou složku proudu $i_{s\alpha}$ a $i_{s\beta}$ prostorového vektoru, jež jsou s tímto systémem spjaty. Základní myšlenkou vektorového řízení je zvolit vhodný souřadnicový systém, ve kterém je možné nezávisle na sobě řídit točivý moment a magnetický tok velikostí složek statorového proudu v ortogonálních souřadnicích, tedy nezávisle na sobě ovládat buzení a moment stroje, jak je tomu u DC motoru s cizím buzením. V případě synchronního motoru s permanentními magnety, u kterého je budící tok vytvářen právě permanentními magnety, se vektorové řízení provádí v souřadné soustavě spojené s polohou rotoru. Požadované složky tak získáme vektorovým natočením reálné a imaginární složky vektoru $i_{s\alpha}$ a $i_{s\beta}$ ze statorového souřadného systému do rotorového systému dq , kde i_{sq} představuje momentotvornou složku a i_{sd} znázorňuje tokotvornou složku prostorového vektoru statorového proudu.

U asynchronního motoru je naopak nejvhodnější zvolit natočení do takového souřadného systému, kde se v reálné ose rotující souřadné soustavy bude nacházet prostorový vektor spřaženého magnetického toku Ψ . Jedná se o orientovaný souřadnicový systém xy spojený s magnetickým polem, kde i_{sx} tvoří budící, tedy tokotvornou složku statorového proudu, zatímco i_{sy} složku momentotvornou. Ve výsledku je spřažený magnetický tok Ψ_x buzený i_{sx} složkou statorového proudu. Pokud bychom srovnali vztahy pro výpočet momentu DC motoru s cizím buzením, který je tvořen součinem budícího magnetického toku $c\phi$ na statoru a kotevního proudu I_a , s výpočtem momentu u asynchronního motoru, zjistili bychom, že nám tento moment stroje nyní tvoří součin konstanty K , spřaženého toku Ψ_s buzeného složkou i_{sx} a momentotvorné složky statorového proudu i_{sy} . Výsledkem těchto transformací je vyjádření střídavých veličin stejnosměrnými a řídit tak střídavý motor se všemi jeho výhodami plynoucími z absence komutátoru stejným způsobem, jako DC motor s cizím buzením.

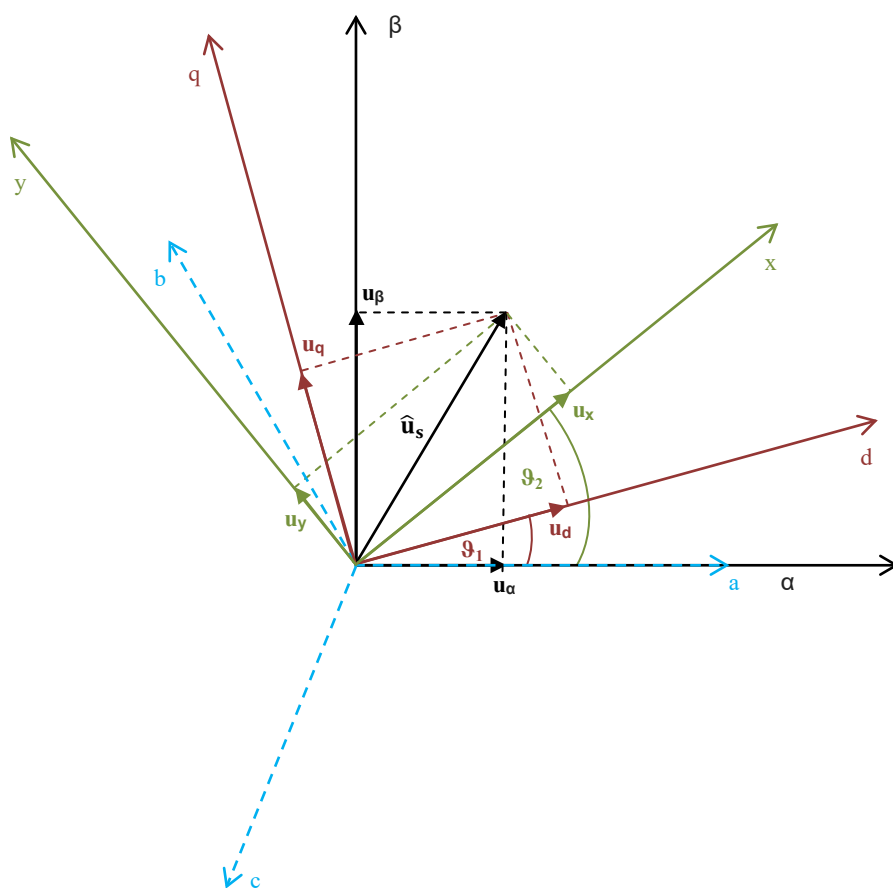
2.6.3 Komplexní souřadnicové systémy [11]

Obecný střídavý stroj můžeme vyjádřit matematickým popisem, u kterého zavedeme komplexní prostorové vektory.

Užíváme tři základní komplexní souřadnicové systémy:

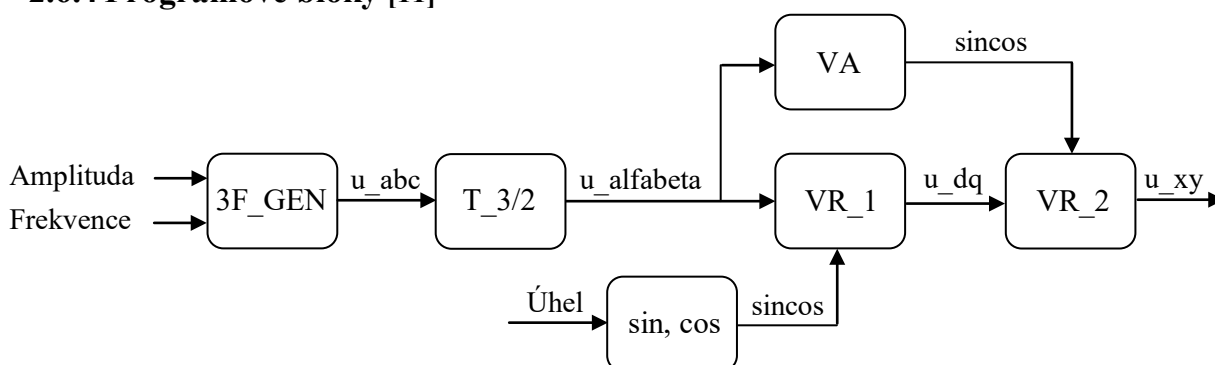
- pevný statorový souřadnicový systém (α, β)
- rotorový souřadnicový systém (d, q) otáčející se úhlovou rychlostí rotoru ω_m
- orientovaný souřadnicový systém (x, y) otáčející se úhlovou rychlostí točivého mg. pole statoru ω_s

Na *obr. 42* můžeme vidět rozložení jednotlivých komplexních souřadnicových systémů. Napětí převedeme do souřadnicového systému xy . Z tohoto systému nyní můžeme provádět vektorové rotace, které nám přepočítají souřadnice našeho napětíového vektoru do dalšího souřadnicového systému. Stejná transformace platí i pro další veličiny, které chceme takto převádět. Tyto operace se provádějí z důvodu zjednodušení návrhu řídicího systému a k minimalizaci počtu rovnic. Návrh řídicího systému v třífázové soustavě by byl v opačném případě mnohem náročnější a prodloužil by výpočetní čas mikropočítače.



Obr. 42: Zobrazení základních komplexních souřadnicových systémů s transformací napětí (podle [11])

2.6.4 Programové bloky [11]



Obr. 43: Blokové zapojení jednotlivých funkčních bloků (podle [12])

Blok 3F_GEN

Pomocí tohoto bloku budeme generovat 3f. průběh napětí pomocí funkce $u = U_m \cdot \sin(\omega t)$. Funkce $\sin(x)$ se bude generovat pomocí tabulky, ve které bude uloženo 256 y -hodnot popisující $\frac{1}{4}$ její periody. Y -hodnoty ostatních částí periody budou rovněž vybírané z téže tabulky, avšak v odlišném pořadí. Signály budou vůči sobě fázově posunuty : $\varphi_A = 0^\circ$, $\varphi_B = 120^\circ$ a $\varphi_C = -120^\circ$.

Blok T_3/2 (Clarkova transformace)

Tento blok bude provádět transformaci 3f signálu (u_a , u_b a u_c) na dvoufázový (u_α , u_β). Výpočet můžeme provádět dvěma následujícími způsoby:

- a) Na výstup uloží transformované souřadnice z třífázového systému do dvoufázového pomocí **dvou fází**:

$$u_\alpha = u_a \quad 2.6.1$$

$$u_\beta = \frac{1}{\sqrt{3}} \cdot u_a + \frac{2}{\sqrt{3}} \cdot u_b \quad 2.6.2$$

- b) Na výstup uloží transformované souřadnice z třífázového systému do dvoufázového pomocí **všech tří fází**:

$$u_\alpha = \frac{2}{3} \cdot u_a - \frac{1}{3} \cdot u_b - \frac{1}{3} \cdot u_c \quad 2.6.3$$

$$u_\beta = \frac{1}{\sqrt{3}} \cdot u_b - \frac{1}{\sqrt{3}} \cdot u_c \quad 2.6.4$$

Blok VR_1 (Parkova transformace)

Blok provádí vektorové natočení reálné a imaginární složky vektoru ze satorového souřadného systému (α, β) do systému rotorového (d, q) .

Pro tuto transformaci jsou potřebné nejen hodnoty v samotném satorovém souřadnicovém systému α, β , ale také okamžitý úhel mezi stojícími a rotujícími souřadnicemi (úhel θ), který zadáváme do bloku \sin, \cos pro výpočet sinus a cosinus zadaného úhlu.

$$u_d = u_\alpha \cdot \cos\theta + u_\beta \cdot \sin\theta \quad 2.6.5$$

$$u_q = u_\beta \cdot \cos\theta - u_\alpha \cdot \sin\theta \quad 2.6.6$$

V případě, že bychom chtěli realizovat inverzní převod reálné a imaginární složky vektoru z rotorového souřadnicového systému do satorového, použijeme následující rovnice:

$$u_\alpha = u_d \cdot \cos\theta - u_q \cdot \sin\theta \quad 2.6.7$$

$$u_\beta = u_d \cdot \sin\theta + u_q \cdot \cos\theta \quad 2.6.8$$

Blok VA

Vektorový analyzátor vypočítává ze dvou složek vektoru ze satorového souřadného systému (α, β) absolutní velikost vektoru a sinus a cosinus úhlu, který vektor svírá se souřadným systémem.

$$|u_{\alpha\beta}| = \sqrt{u_\alpha^2 + u_\beta^2} \quad 2.6.9$$

$$\sin = \frac{u_\beta}{|u_{\alpha\beta}|} \quad 2.6.10$$

$$\cos = \frac{u_\alpha}{|u_{\alpha\beta}|} \quad 2.6.11$$

Blok VR_2 (Parkova transformace)

Blok nyní provádí vektorové natočení reálné a imaginární složky vektoru z rotorového souřadného systému (d, q) do systému orientovaného (x, y) podle úhlu, který jsme získali pomocí vektorové analýzy.

$$u_x = u_d \cdot \cos\theta + u_q \cdot \sin\theta \quad 2.6.12$$

$$u_y = u_q \cdot \cos\theta - u_d \cdot \sin\theta \quad 2.6.13$$

V případě, že bychom chtěli realizovat inverzní převod reálné a imaginární složky vektoru z orientovaného souřadnicového systému do rotorového, použijeme následující rovnice:

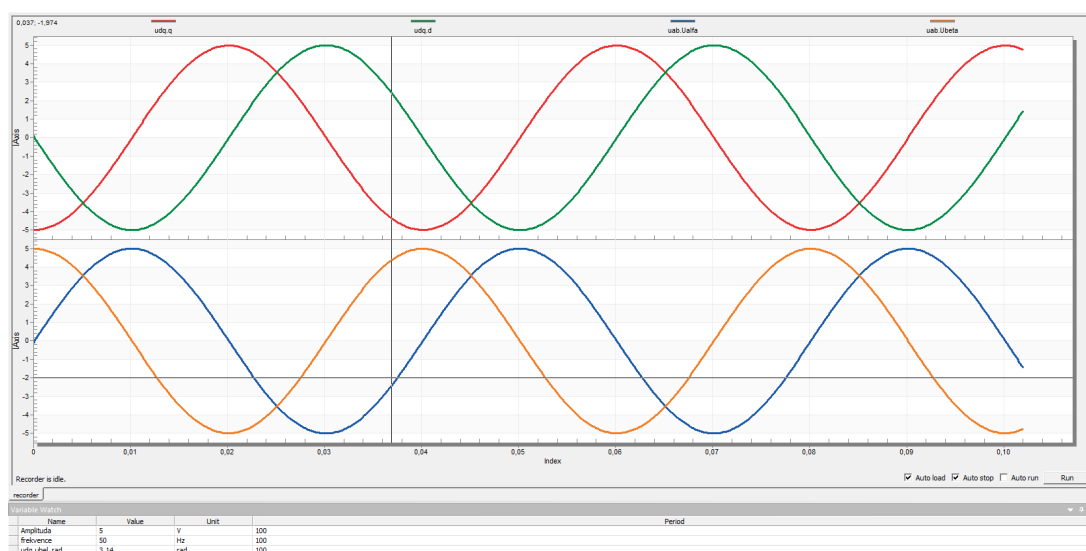
$$u_d = u_x \cdot \cos\theta - u_y \cdot \sin\theta \quad 2.6.14$$

$$u_q = u_x \cdot \sin\theta + u_y \cdot \cos\theta \quad 2.6.15$$

2.6.5 Dosažené výsledky

Generovat 3fázový signál, jak již bylo zmíněno, je možné třemi způsoby: dle tabulky pro $\frac{1}{4}$ periody sinus funkce, Taylorova polynomu nebo dle funkcí, které obsahuje již předpřipravená knihovna *math.h*, mezi nichž patří mj. sinus (*sin*), cosinus (*cos*), odmocnina (*sqrt*) či absolutní hodnota (*abs*). Funkci *sqrt* a *abs* je však možné realizovat samostatně pomocí jednoduchých pravidel, které si uživatel během vývoje programu osvojí.

V první části následujícího obrázku můžeme vidět 2fázový signál se složkami u_α , u_β prostorového vektoru, který se nachází ve statorovém souřadném systému. Druhá část znázorňuje vektorovou rotaci, při které dojde k převedení prostorového vektoru do rotorového souřadného systému se složkami d , q s natočením o zadaný úhel, který v našem případě činí π rad.



Obr. 44: Snímek vektorové rotace 2fázového signálu ze SSS (u_α , u_β) do RSS (d , q) dle úhlu $\varphi = 3,14$ rad ve FreeMasteru

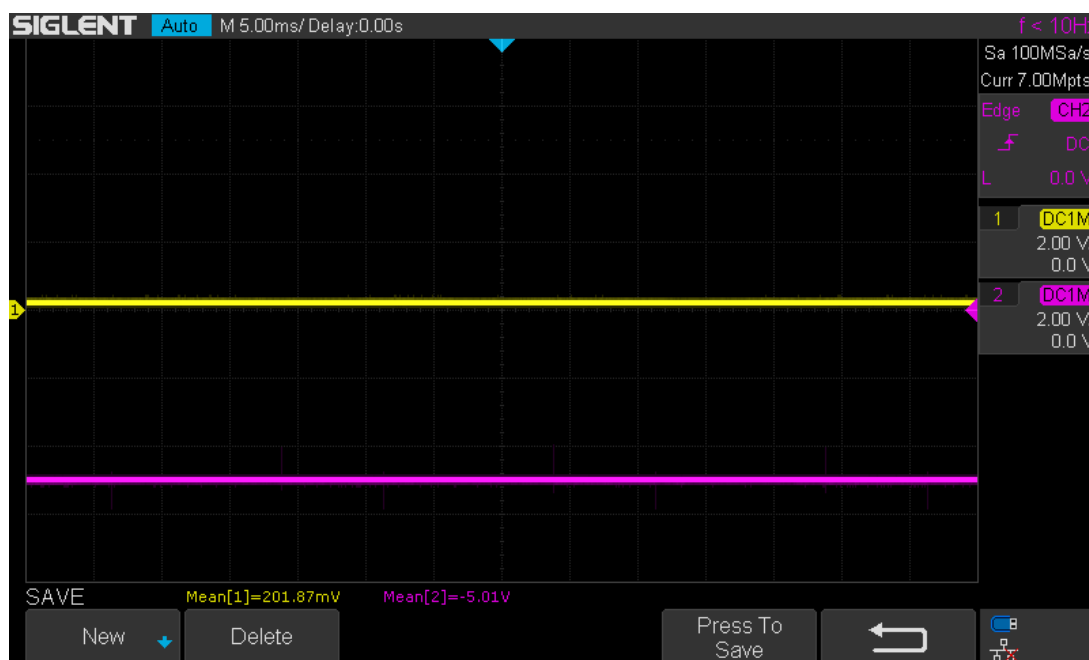
Příklad realizace matematických funkcí

```
float abs_f(float x) { /* vypočet absolutní hodnoty */
    if(x < 0.0)
        return -x;
    else
        return x;
}

float odmocnina(float x) { /* vypočet odmocniny */
    float y, n, presnost;
    n = x;
    y = n * 0.5;
    presnost = 0.000001; /* presnost */

    while(abs_f(x - y) > presnost) {
        x = (x + y) * 0.5;
        y = n / x;
    }
    return(x); /* vraceni vysledku odmocniny */
}
```

Na obr. 45 je pořízený snímek z osciloskopu zobrazující vektorovou rotaci z rotorového souřadného systému do orientovaného (x, y), kdy reálná stejnosměrná složka (x) prostorového vektoru při natočení o $\pi \text{ rad}$ nabývá hodnoty -5 , zatímco imaginární složka (y) je nulová.



Obr. 45: Zobrazení vektorové rotace z RSS (d, q) do OSS (x, y) dle úhlu získaného pomocí VA ve FreeMasteru, $\varphi = \pi \text{ rad}$

Příklad realizace některých funkčních bloků

```
/* Podprogram pro vektorovou analýzu v daném souřadném systému */
void va_SSS(SSS *uab, UHEL *scp) {
    scp->prepona = sqrt((uab->Ualfa*uab->Ualfa) + (uab->Ubeta*uab->Ubeta));
    scp->sin_z = uab->Ubeta/scp->prepona;
    scp->cos_z = uab->Ualfa/scp->prepona;
}
```

```
/* Podprogram pro vektorové natocení z RSS do OSS */
void vr_RSS_do_OSS(RSS *udq, OSS *uxy, UHEL *scp) {
    uxy->x = udq->d*scp->cos_z + udq->q*scp->sin_z;
    uxy->y = udq->q*scp->cos_z - udq->d*scp->sin_z;

    #if DAC01_vyber_zobrazeni == 4
    Ux_DAC0 = 204.7 * uxy->x + 2047;
    Uy_DAC1 = 204.7 * uxy->y + 2047;

    DA0_SetValue16(&Ux_DAC0);
    DA1_SetValue16(&Uy_DAC1);
    #endif
}
```

3 Vypracované protokoly

3.1 Zásady pro vypracování protokolů

Na správně zpracovaný protokol se vztahují určité zásady, které je nezbytné dodržet. Patří mezi ně mimo jiné zadání obsahující všechny požadované části, které je student povinen v dané úloze splnit. Vlastně zpracovaný teoretický rozbor, který patří mezi naprostý základ každého protokolu, má dopomoci k tomu, aby se student s daným tématem své práce patřičně seznámil a porozuměl principům dílčích částí a uvědomil si tak, co se po něm požaduje. Ve vzorových protokolech této závěrečné práce je teoretický rozbor vynechán záměrně, neboť tento rozbor je obsahem podrobných manuálů sestavených k jednotlivým laboratorním úlohám, které jsou jednou z příloh mé práce.

Příloha v podobě programového kódu sestaveného programu, či alespoň jeho část (zdroj. soubor *Aplikace.c*), je pro zpracovaný protokol nepostradatelná. Tento programový kód je do protokolu možné přikládat různými způsoby, jakými jsou např. funkce *Print* ve vývojovém prostředí KDS, nebo *screenování*. Z mého pohledu se však jako nejvhodnější způsob vygenerovaného programu, který je možné v patřičném čitelném formátu přiložit, jeví kombinace prostředí programu KDS a Visual Studia od společnosti *Microsoft*. Část sepsaného programu nacházejícího se ve zdrojovém souboru *Aplikace.c* je totiž možné otevřít pomocí Visual Studia, který mimo jiné nabízí bohaté možnosti úpravy barev a stylů dílčích částí programu (deklarace proměnných, typy, funkce, podprogramy, text, ...) a je schopný čitelně zpracovat poznámky programu dosahujících dlouhých rozměrů.

Pro ověření správné funkčnosti napsaného programu je vhodné obohatit protokol o snímky z osciloskopu (výstupní průběh na připojené LED, generované funkce $\sin(x)$ a $\cos(x)$, reg. zásah, vektorové rotace, ...) a samozřejmě také z graf. nástroje *FreeMasteru*, který je nenahraditelným pomocníkem při vývoji našich aplikací.

Závěr je většinou konečný bod protokolů. Správně by měl obsahovat stručné popsání jednotlivých částí dotyčného programu, shrnout a popřípadě porovnat, vyhodnotit a objasnit dosažené výsledky. Za závěr se v určitých okolnostech přidávají citace, které se řídí normou ISO-690, popř. ISO-690-2 a to v případě, že je uvedený teoretický rozbor čerpán ze zdrojů, které nejsou našim dílem.

4 Podpůrné materiály

4.1 Funkce jednotlivých komponent vývojového nástroje PE

V této kapitole budou vyjmenovány všechny funkce nabízené nástrojem PE, které je možné použít pro vývoj výše zmíněných aplikací.

1. Laboratorní úloha č. 1

void LED1_Neg(void){ }

Funkce komponenty *LED*, která nedisponuje vstupním parametrem a také žádný parametr nevrací, neguje stav LED, tedy uvádí LED z log. 1 do log. 0 a naopak.

void WAIT1_Waitms(uint16_t ms){ }

Tuto funkci nabízí komponenta *Wait*. Vstupním parametrem je 16bit. celočíselná proměnná *ms*, jejíž hodnota definuje velikost zpoždění v jednotkách ms.

2. Laboratorní úloha č. 2

(void) FMSTR_Poll(void){ }

Jedná se o jedinou potřebnou funkci pro náš program a přitom zcela klíčovou pro komunikaci s grafickým nástrojem PE. Funkce se nachází pod komponentou *FreeMaster* a slouží pro udržení komunikace s tímto nástrojem během provádění našeho programu. Zpracovává nám dekódování a provádění protokolu komunikace. Jedná se o důležitou funkci, která se zpravidla umísťuje do nekonečné smyčky programu. Bez této funkce v této smyčce s námi *FreeMaster* nebude komunikovat. Funkce nemá vstupní parametry a nic nevrací.

3. Laboratorní úloha č. 3

Funkce *FMSTR_Poll()*; se bude také nacházet v programech zmíněných níže, pro které je rovněž zcela nezbytná, avšak z důvodu, že již byla podrobně rozebrána výše, již nadále nebude zmíněna.

#define DA0_SetValue16(Values) (byte)DacLdd1_SetValue(DacLdd1_DeviceData, (LDD_DAC_TData)((word*)(Values)))*

Funkce je definovaná jako makro s názvem *DA0_SetValue16* se vstupní hodnotou proměnné *Values*, která je definovaná jako pointer ukazující na adresu proměnné, ve které je uložena 16bit. napěťová hodnota. Tato funkce tedy převádí max. 16bitovou digitální hodnotu na analogovou veličinu, kterou lze následně za pomoci nástrojů FM sledovat v čase.

(void) FMSTR Recorder(void){ }

Funkce nám vzorkuje proměnné v *Recorderu*. *Recorder* (záznamník) je „vytvořen v softwaru na cílové vývojové desce, kde ukládá změny proměnných v reálném čase“. Zvolené proměnné v okně *Recorderu* ve FM jsou periodicky zaznamenávány pomocí přerušení vestavěného časovače. Po dosažení nastaveného počtu vzorků těchto proměnných jsou tyto vzorky uloženy uvnitř vyrovnávací paměti *Recorderu* umístěné na cílové desce, ze které jsou následně staženy a nakonec zobrazeny v okně tohoto záznamníku. [13]

4. Laboratorní úloha č. 4

byte AD1 Measure(bool WaitForResult){ }

Funkce spouští měření A/D převodníku. Převodník vzorkuje analogový signál na vstupu, ze kterého získává analogovou hodnotu. Takto navzorkovanou hodnotu převádí do 16bit. slova, které ukládá do paměti převodníku. V případě zvolení více kanálů převodníku dochází k jejich multiplexování, a tak se výše zmíněný postup vzorkování, převádění a ukládání hodnot opakuje pro každý následující kanál, na který se přivádí odlišný analog. signál. To má však za následek zvýšení doby převodu.

*byte AD1 GetChanValue16(byte Channel, word *Value){ }*

Pomocí této funkce vybíráme uloženou 16bit. hodnotu z paměti převodníku, kterou následně ukládáme do zvolené proměnné nebo pole. Jedním ze vstupních parametrů funkce je výběr kanálu převodníku, ve kterém je převedená hodnota uložena. Alternativou je použití funkce *AD1_GetValue16()*, která nám zajistí přesunutí a uložení všech hodnot z paměti převodníku do námi zvoleného pole bez nutnosti výběru přesného kanálu.

5. Laboratorní úloha č. 5 a 6

Při vývoji těchto programů se používají totožné funkce komponent *ADC*, *DAC* a *FreeMaster*, které již byly zmíněny výše.

4.2 Dokumentace složitějších funkcí

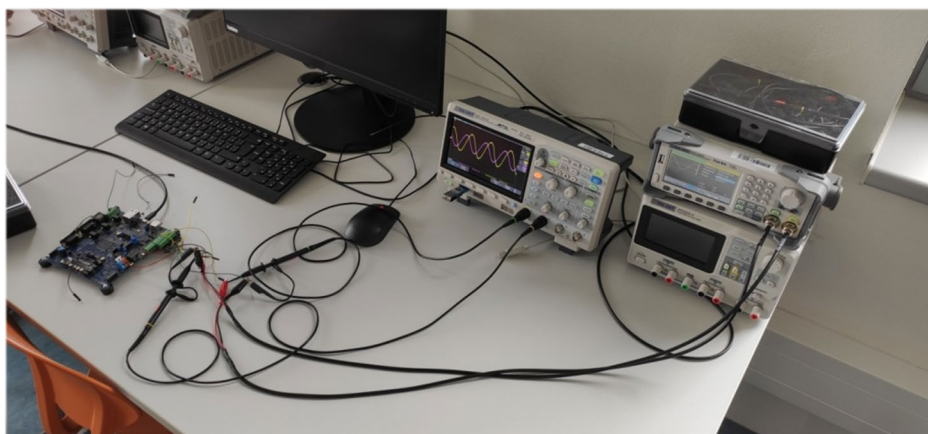
Pro zjednodušení práce při vývoji daných aplikací byla v rámci bakalářské práce vytvořena dokumentace tvořící zdrojový a hlavičkový soubor složitějších funkcí, které budou nezbytné pro úspěšnou tvorbu programů. Soubor s dokumentací se rovněž nachází v příloze této práce.

Závěr

Cílem závěrečné práce bylo učinit vše proto, aby se studijní předmět MŘS2 stal dostatečně pochopitelným, ne-li v řadě studentů oblíbeným. Pro tento předmět byly sepsány obsáhlé manuály, ve kterých se nachází jak zadání s teoretickým rozбором, tak také podrobný postup řešení, s jehož pomocí by studentům nemělo činit nadměrné obtíže orientovat se a pracovat na jednotlivých úlohách ve zmíněných programech. Pro každou laboratorní úlohu byly sepsány a vytvořeny vzorové protokoly, jež poukazují na správnost dosažených výsledků a jejichž závěry jsou věnovány podrobnému popisu funkčnosti daného programu.

Tato práce mi pomohla porozumět významu a potenciálu, kterým disponují dnešní mikropočítače a bez kterých by se svět, který známe dnes, jen stěží obešel. Mikropočítače se v současné době nacházejí ve všech zařízeních, na která si jen dokážeme vzpomenout. Disponují nespočetnými možnostmi realizace, jejich možnosti jsou „odemčeny“ daným výrobcem a dovedou si poradit s každou nevšední situací ať již v těžkém průmyslu (řízení a regulace veličin motorů), nebo v domácím prostředí.

Mezi počítačovými programy, které byly využity pro zpracování této bakalářské práce, se řadí vývojové prostředí *Kinetis Design Studio* s doplňkem *Processor Expert* v řadě případů ulehčující vývoj aplikací bez nutnosti hluboké znalosti programování mikrokontrolérů a také grafický nástroj *FreeMaster*, který se osvědčil jako nepostradatelný pomocník při ověřování správnosti řešení napsaného programu.



Obr. 46: Zobrazení používaných přístrojů k měření a testování aplikací

Přestože se tato práce obsáhle zabývala generováním, zpracováním, řízením a regulací signálů a veličin, mnoho možností vývojové desky a tak samotného mikrokontroléru nebylo využito. Jako nastavbovou úlohu bych navrhl využití Ethernetu, s jehož pomocí by na desce bylo možné provozovat jednoduchý webový server, na který by bylo možné nechat si zasílat zaznamenané informace z připojených či integrovaných čidel na desce, např. údaje o stavu či teplotě prostředí. Dále by bylo možné do aplikací zapojit sběrnici CAN nacházející široké uplatnění v automobilovém průmyslu při zajištění vzájemné komunikace mezi větším počtem řídicích jednotek.

Literatura


- [1] ELECTRONICS | 69, Circuit Basics | DIY. Basics of UART Communication. *Circuit Basics* [online]. 13. únor 2016 [vid. 2020-03-29]. Dostupné z: <https://www.circuitbasics.com/basics-uart-communication/>
- [2] MAZIDI, Muhammad Ali, Sarmad NAIMI, Sepehr NAIMI a Shujen CHEN. *Freescale ARM Cortex-M embedded programming: using C language*. B.m.: Microdigitaled, 2016. ISBN 978-0-9979259-8-2.
- [3] KOUŘIL, Daniel. *Vývojová deska K64 - dokumentace, ver. 1.1*. B.m.: VŠB-TU Ostrava, Katedra elektroniky (430)
- [4] VYSLOUŽIL, Tomáš. Taylorův polynom. *GeoGebra* [online]. [vid. 2020-03-30]. Dostupné z: <https://www.geogebra.org/m/KnxtJbK9>
- [5] PRAUZEK, Michal. Vestavěné řídicí systémy 2016/2017 - Přednáška 4. In: [online]. VŠB-TU Ostrava. [vid. 2020-03-30]. Dostupné z: https://homel.vsb.cz/~pra132/vrs/VRS_prednaska4.pdf
- [6] *Sample and Hold Circuit Diagram* [online]. [vid. 2020-03-30]. Dostupné z: <https://circuitdigest.com/electronic-circuits/sample-and-hold-circuit-op-amp>
- [7] PAVELEK, Tomáš. *Technické prostředky pro řízení elektrických pohonů (TPŘEP) - Návod do cvičení: PI regulátor* [online]. B.m.: VŠB-TU Ostrava, Katedra elektroniky (430). [vid. 2020-03-30]. Dostupné z: https://lms.vsb.cz/pluginfile.php/849706/mod_resource/content/0/PI%20regul%C3%A1tor.pdf
- [8] SOBEK, Martin. *Mikropočítačové řídicí systémy II - Laboratorní úloha 4: Bloky regulačních systémů*. B.m.: VŠB-TU Ostrava, Katedra elektroniky (430)
- [9] ŠTĚPANEK, Libor. *Elektrické regulované pohony I (ERP1) - Přednáška A7 až A10: Regulace el. pohonů - Stabilita a tlumení* [online]. B.m.: VŠB-TU Ostrava, Katedra elektroniky (430). [vid. 2020-03-30]. Dostupné z: https://lms.vsb.cz/pluginfile.php/1171105/mod_resource/content/1/A07%20a%C5%BE%2010_ERP1-regulace_EP%202017.pdf
- [10] BRANDŠTETTER, Pavel. *Technické prostředky pro řízení elektrických pohonů (TPŘEP) - Učební texty pro kombinované a distanční studium*. B.m.: VŠB-TU Ostrava, Katedra elektroniky (430). 2005
- [11] BRANDŠTETTER, Pavel. *Mikropočítačové řídicí systémy II. Učební texty pro kombinované a distanční studium*. Ostrava: VŠB-TU, 2010. ISBN xxxxxxxxx.
- [12] SOBEK, Martin. *Mikropočítačové řídicí systémy II - Laboratorní úloha 3: Programové bloky pro řízení pohonů*. B.m.: VŠB-TU Ostrava, Katedra elektroniky (430)
- [13] *FreeMASTER for Embedded Applications: User Guide* [online]. [vid. 2020-03-29]. Dostupné z: <https://www.nxp.com/docs/en/user-guide/FMSTRUG.pdf>
- [14] *Referenční manuál MK64FN1M0VLQ12* [online]. [vid. 2020-03-28]. Dostupné z: <https://www.mouser.com/datasheet/2/813/K64P144M120SF5RM-1074828.pdf>

- [15] Basics of UART Communication. *Electronics Hub* [online]. 17. červen 2017 [vid. 2020-03-29]. Dostupné z: <https://www.electronicshub.org/basics-uart-communication/>
- [16] SOBEK, Martin. *Mikropočítačové řídicí systémy II - Laboratorní úloha 1: Komunikace s uživatelem*. B.m.: VŠB-TU Ostrava, Katedra elektroniky (430)
- [17] MAŘÍK, Robert. *Taylorův polynom* [online]. [vid. 2020-03-30]. Dostupné z: <http://user.mendelu.cz/marik/mat-web/mat-webse4.html>
- [18] PRAUZEK, Michal. *ČÍSLICOVÁ A MIKROPROCESOROVÁ TECHNIKA*. Ostrava: VŠB-TU, 2013. ISBN xxxxxxxxx.
- [19] *Cookbook for SAR ADC Measurements* [online]. [vid. 2020-03-30]. Dostupné z: <https://www.nxp.com/docs/en/application-note/AN2371.pdf>
- [20] Average Voltage of a Sinusoidal AC Waveform. *Basic Electronics Tutorials* [online]. 25. červen 2013 [vid. 2020-03-30]. Dostupné z: <https://www.electronics-tutorials.ws/accircuits/average-voltage.html>
- [21] RMS Voltage of a Sinusoidal AC Waveform. *Basic Electronics Tutorials* [online]. 25. červen 2013 [vid. 2020-03-30]. Dostupné z: <https://www.electronics-tutorials.ws/accircuits/rms-voltage.html>
- [22] TAUSIF AHMAD SHEKH, Muktar, Umale ANKIT R. a Kirpane RAHUL K. *Vector control methods for variable speed of ac motors* [online]. 2017 [vid. 2020-03-30]. Dostupné z: <https://www.irjet.net/archives/V4/i2/IRJET-V4I267.pdf>







Přílohy

V přílohách se nacházejí zpracované manuály k jednotlivým laboratorním úlohám, jejichž obsah mj. tvoří podrobné postupy k práci s aplikačním softwarem od společnosti NXP a také samotná zadání úloh.

_Manuály k laboratorním úlohám







-  Lab00_[MRS2].pdf
-  Lab01_[MRS2].pdf
-  Lab02_[MRS2].pdf
-  Lab03_[MRS2].pdf
-  Lab04_[MRS2].pdf
-  Lab05_[MRS2].pdf

_Samotná zadání LÚ

-  Lab00_[MRS2]_Zadání.docx
-  Lab01_[MRS2]_Zadání.docx
-  Lab02_[MRS2]_Zadání.docx
-  Lab03_[MRS2]_Zadání.docx
-  Lab04_[MRS2]_Zadání.docx
-  Lab05_[MRS2]_Zadání.docx







Nechybí také sestavené a ověřené programy.

_Programy

-  LAB00_[MRS2]
-  LAB01_[MRS2]
-  LAB02_[MRS2]
-  LAB03_[MRS2]
-  LAB04_[MRS2]
-  LAB05_[MRS2]

Poslední bod zadání je také splněn za pomoci vypracovaných protokolů ke všem laboratorním úlohám.

_Vypracované protokoly

-  Laboratorní úloha 0.pdf
-  Laboratorní úloha 1.pdf
-  Laboratorní úloha 2.pdf
-  Laboratorní úloha 3.pdf
-  Laboratorní úloha 4.pdf
-  Laboratorní úloha 5.pdf

Dokumentace složitějších funkcí:

-  Funkce.c
-  Funkce.h